REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)*
15-03-2005

2. REPORT TYPE
Final

01-05-2001 to 31-10-2004

4. TITLE AND SUBTITLE
Multimodal Human Identification for Computer Security

5a. CONTRACT NUMBER

5b. GRANT NUMBER
F49620-01-1-0343

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Sohail Nadimi, Edward Hong and Bir Bhanu

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Center for Research in Intelligent Systems
Bourns College of Engineering
University of California
Riverside, CA 92521

8. PERFORMING ORGANIZATION REPORT NUMBER
2005-10

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFOSR
4015 Wilson Blvd.
Arlington, VA 22203-1954

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Unlimited

DISTRIBUTION STATEMENT A
Approved for Public Release
Distribution Unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes the work perforemd by CIPIAF fellows. Three research ideas are presented:

(a) A cooperative coevolutionary approach for object detection is developed. It fuses the scene contextual information with the available statistical and prediction information available from color and infrared sensors. The sensor fusion system maintains high detection rates under a variety of environmental conditions. The results are shown for a full 24 hour diurnal cycle.

(b) An agent-based intrusion detection system, where evolutionary computational techniques, similar to those discussed in (a) are explored. A detailed architecture for a coevolutionary agent based system is given and the concept of super agent is described.

(c) A performance modeling approach for object recognition is developed and the results are shown on synthetic aperture radar images.

15. SUBJECT TERMS
Evolutionary Computation, Adaptation and Learning, Intrusion Detection, Performanec Modeling, Object Detection, Object Recognition.

16. SECURITY CLASSIFICATION OF: Unclassified			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Spencer Wu
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**		85	**19b. TELEPHONE NUMBER** *(include area code)* 703-696-7315

Table of Contents

Chapter 1. Introduction

This is the final report on Multimodal human identification for computer security, AFOSR grant No. F49620-01-1-0343. The grant was part of a Critical Infrastructure Protection and Information Assurance Fellow (CIPIAF) Program to support one fellow. The PI on this Program was Prof. Bir Bhanu, Center for Research in Intelligent Systems, University of California, Riverside.

Initially, we hired Dr. Ed Hong (U.S. citizen), who has a B.S. in Computer Science and Math from Yale and M.S. and Ph.D. in Computer Science (2002) from the University of Washington. Dr. Hong has a very strong background in mathematics and computer science. He conducted a focused research project on analytical methods for predicting the fundamental performance of recognition systems. Dr. Hong developed an improved method for predicting the bounds on performance of a vote-based object recognition system, when the test data features are distorted by uncertainty in both feature locations and magnitudes, by occlusion and by clutter. The method calculates lower and upper bound predictions of the probability that objects with various levels of distorted features will be recognized correctly. The prediction method takes model similarity into account, so that when models of objects are more similar to each other, then the probability of correct recognition is lower. The effectiveness of the prediction method was validated in a synthetic aperture radar (SAR) automatic target recognition (ATR) application using MSTAR public SAR data, which are obtained under different depression angles, object configurations and object articulations. Experiments show the performance improvement that can be obtained by considering the feature magnitudes, compared to a previous performance prediction method that only considered the locations of features.

Dr. Hong left UCR at the beginning of the fall 2003 for a full time tenure track faculty position in the Department of Computer Science at the University of Washington.

The key accomplishments of Dr. Hong have been (a) writing a conference paper and (b) presenting it at the SPIE conference in April 2003. Since then he has done some additional theoretical and experimental work and a draft is under revision which will be submitted for journal publication. The conference paper presented by Dr. Hong is provided as Chapter 4 in this report.

After Dr. Hong's departure, in June 2003 we hired Dr. So hail Madame (U.S. citizen) as the Fellow on this grant. Dr. Nadimi received his B.S. in Industrial Technology and M.S. in Computer Science from San Jose State University and the Ph.D. degree in Computer Science (2003) from the University of California at Riverside. Prior to pursuing his Ph.D. degree he worked at IBM Almaden Research Center for four years. Dr. Nadimi's research interests are in computer vision, machine learning, image processing, image and pattern recognition and artificial intelligence.

Dr. Nadimi worked on two major tasks:

1) An evolutionary sensor fusion system for object detection 24 hours a day. It is described in Chapter 2. This chapter has been submitted for publication as a book Chapter in a book entitled, "Optical Imaging, Photonics, Sensors, and Systems for Homeland Security," (B. Javidi, Editor), Springer (to be published). An expanded journal paper will be published in the future.

2) An agent-based intrusion detection system where evolutionary based techniques, similar to the one developed in 1) are explored. A detailed architecture for a coevolutionary agent based system is given and the concept of a super agent is developed. New features are developed that can be used for intrusion detection. The research performed under this task is described in detail in Chapter 3. Dr. Nadimi left UCR in October 2004 and has taken a position with KLA-Tencor in San Jose, California.

Chapter 2. Evolutionary Sensor Fusion for Security

Abstract: A robust moving object detection system for an outdoor scene must be able to handle adverse illumination conditions such as sudden illumination changes or lack of illumination in a scene. This is of particular importance for scenarios where active illumination cannot be relied upon. Utilizing infrared and video sensors, we develop a novel sensor fusion system that automatically adapts to the environmental changes that affect sensor measurements. The adaptation is done through a cooperative coevolutionary algorithm that fuses the scene contextual and statistical information through a physics-based method. The sensor fusion system maintains high detection rates under a variety of conditions. The results are shown for a full 24 hour diurnal cycle.

2.1. Introduction

Over the past several decades many approaches have been developed for moving object detection for indoor and outdoor scenes. Moving object detection methods fall into two categories: (a) feature-based methods [21], and (b) featureless methods (e.g., image subtraction, optical flow, statistical modeling) [2, 4, 6, 8, 18, 24]. Each of these methods offers advantages that are exploited for different applications. For example, temporal differencing is simple and may suffice for indoor type illuminations for slow moving objects, optical flow is useful for a moving camera platform and statistical modeling can capture the background motion.

Some of the shortcomings of the above approaches for moving detection are:

1) None of these approaches address the problem of low light or no light conditions,

2) No contextual information is used to update the parameters,

3) Generally, a large number of observations are required before a background model can be learned effectively,

4) The algorithms have been applied to a single sensing modality (usually visible or near-infrared) and no results have been shown for extreme conditions, for example, no illumination, sunset, or sunrise condition.

To overcome illumination conditions such as low or no light conditions, other sensing modalities such as cameras operating in near or longwave IR have been utilized [3]. However, these sensing modalities could still fail due to similar conditions in their respective bandwidth. For example, in a longwave (thermal Infrared) camera, a subject's temperature could reach that of the background, thus having limited contrast which may cause detection failure.

Multisensor fusion attempts to resolve this problem by incorporating benefits of different sensing modalities. The advantages of multisensor fusion are improved detection, increased accuracy, reduced ambiguity, robust operation, and extended coverage. Sensor fusion can be performed at different levels including signal or pixel level, feature level and decision level.

This chapter provides a novel sensor fusion system that fuses longwave (thermal IR) and visible sensors in a unified manner. By utilizing the IR signal, we can overcome some of the limitations of the visible cameras and by combining the visible and IR signal we improve the detection under a variety of conditions. The salient features of our approach are:

a) *Consistent data representation:* At the image level all sensing modalities are represented by mixture of Gaussians in a consistent manner.

b) *Physical models*: Sound physical models are used for each sensing modality (e.g., visible and IR) to provide prediction for each signal.

c) E*volutionary-based approach for fusion*: A cooperative coevolutionary algorithm is developed to systematically fuse and integrate information from both statistical and physical models into a unified structure for detection.

d) C*ontext-based adaptation:* Environmental conditions such as ambient air temperature, wind velocity, surface emissivity, etc., are directly incorporated into the detection algorithm and influence the fusion strategies.

Chapter 2.2 provides the related work and motivation, Chapter 2.3 presents the details of the technical approach, Chapter 2.4 discusses the experimental results and finally Chapter 2.5 provides the conclusions of the chapter.

2.2. Related Work and Motivation

Current multisensor fusion and integration approaches use the following paradigms:

(a) Statistical Paradigm: This paradigm utilizes the statistical properties of signal at pixel, feature or decision level. It includes statistical methods such as Bayesian, Dempster-Shafer, and Fuzzy approaches. These approaches have been used extensively for fusion due to their well-developed mathematics. In [1] Bayesian and Dempster-Shafer multisensor fusion methods are compared for target identification. In [9] a Bayesian-based method for lane detection is developed. In [15, 16] statistics-based techniques have been used for fusing video, near infrared (NIR), mid-wave infrared (MWIR) and long wave infrared (LWIR) signals for image enhancement. Statistical-based fusion approaches provide a unified framework and methods that can deal with sensor noise; however, they require enormous amounts of data and prior knowledge of statistical properties of the signals.

(b) Artificial Intelligence (AI) paradigm: This paradigm attempts to fuse the data through methods such as knowledge-based, rule-based and information theoretic methods. Examples of

AI-based fusion techniques are [5, 20] for image enhancement and target detection, and [7] for robotics. This paradigm has the advantage of incorporating contextual information, heuristics and domain knowledge by utilizing well developed algorithms in the AI field; however, once designed, addition of new sensing modalities requires a new set of algorithms and/or domain knowledge and heuristics that are generally provided by external experts for expansion of knowledge rules.

(c) Data Structure paradigm: This paradigm utilizes various representations such as graphs, trees, tables and data structure-specific techniques such as graph traversal. Terrian [19] and Waxman [23] provide methods for fusing FLIR and an image intensifier data for image enhancement. In [10] an approach is introduced to fuse acoustic and video data for underwater vehicle tracking. This paradigm works well when the data can be represented by one of the structures mentioned. The obvious disadvantage of this paradigm is that once the data structure is defined, it may not be possible to extend the method to new sensing modalities; therefore, this paradigm is suitable when all sensing modalities participating in fusion are known in advance.

(d) Physics paradigm: This paradigm utilizes the sensor phenomenology to model the signals, based on the physical aspect of the world. Physical models describe the relation of object parameters (e.g., surface reflectance, orientation, roughness, temperature, material density, etc.) to scene environmental parameters (such as ambient temperature, direction of illumination, wind velocity, etc.) to predict sensor values.

Pavlidis et al. [13] develop an automatic passenger counting system based on sub-bands below short wave infrared (SWIR). They measure reflectance of many objects including human beings; they note that the human skin reflectance spectral map is very similar to that of distilled water;

they relate this phenomenon to the fact that humans are 70% water. In [12] several physical models have been developed to model the thermal, acoustic and laser radar signals for various segmentation problems. The fusion is viewed as the problem of relating scene parameters to object parameters. Since IR bands above 3μm increasingly measure thermal fluctuations, they model a surface based on heat conductance and use the conservation of energy to model the interaction of surface and radiation.

Among the four paradigms AI and data structure-based paradigms are less suited for dynamic conditions whereas the statistics and physics-based paradigms are the methods of choice for integrating sensor information that can change over time. We provide a new sensor fusion technique that combines the statistical and physics-based fusion paradigms through an evolutionary process. We overcome the disadvantage of each of these paradigms by including suitable sensor models that have enormous generalizing power. This generalizing power is then used to complement the limited available sensor data that is required by the statistical methods. The fusion is performed at the pixel level where the information loss is minimal.

2.3. Technical Approach

The sensor fusion architecture for moving object detection is depicted in Figure 1. Observations from the sensors along with the external conditions, which carry the contextual information, are used to build statistical (mixture of Gaussian) background model. The contextual information is also used to update values of internal physical models. Physical models include reflectance models for predicting image intensity values and thermal models for predicting background surface temperature values. Unlike the previous work that updates the background models solely based on the current observations, we incorporate the physical models into the adaptive loop. The physical models are integrated with the statistical models through a cooperative coveolutionary process [14]. The cooperative coevolutionary process estimates the best representation for the background per pixel. This is done through a genetic evolutionary process that searches for the optimal representation based on the current, and recent past observations and detection results in addition to the predictions given by the physical model.

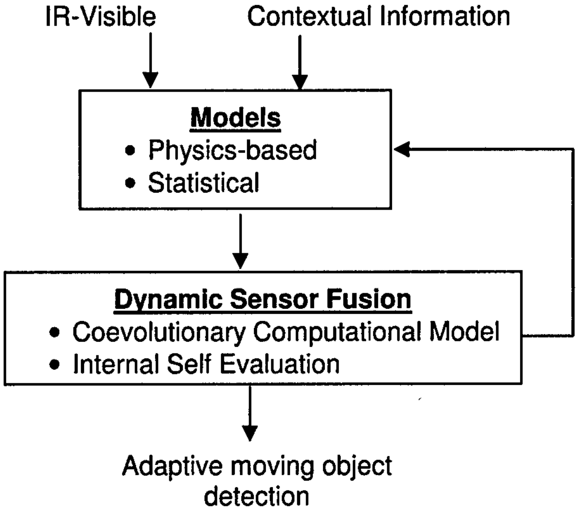

IR-Visible
Contextual Information
Models
• Physics-based
• Statistical
Dynamic Sensor Fusion
• Coevolutionary Computational Model
• Internal Self Evaluation
Adaptive moving object detection

Figure 1. Sensor Fusion Architecture.

Our representation of mixture of Gaussians (described in section 3.1) includes Gaussian parameters for the infrared and visible sensors (including RGB channels). A population of this representation is maintained as a pool of individuals for the evolutionary process. Once the evolutionary process is stopped, the best individual represents the background model of that pixel. In this manner, the contextual information plays an active role in contributing to the most ideal sensor for a particular condition.

The detection algorithm in Figure 1 requires a model of the background. This model is estimated by a mixture of Gaussians. Table 1 shows this process. The details are explained in the following sub-sections.

Table 1. Algorithm for learning background model for a pixel.

```
Evolutionary Adaptive Background Modeling
S = Training set which includes prediction, observation, and previous classification results per pixel;
Note: An organism represents a solution.

---------------------------------- Cooperative Coevolution Algorithm per Pixel----------------------------

Steps
Create and initialize 4 subpopulations for each channel
Loop
  For each Sub-population
    For each individual
      Build an organism (e.g., combine representative individuals from different sub-populations)
      Evaluate the organism using the training set S and $F_{organism}$
      Store the new fitness value for the individual
    EndFor
  EndFor
  Evolve all sub-populations (Selection, Mutation, Crossover)
Until stop Condition
Return the best organism (best organism or solution is the best individual from each subpopulation)
```

2.3.1 Representation

The probability of a pixel, classified as a background, drawn from a probability distribution can be estimated by a mixture of density functions. Assuming the parametric form of the mixture is Gaussian, probability of observing a background pixel is:

$$P(X) = \sum_{i=1}^{m} W_i \, \eta(X, \mu_i, \Sigma_i)$$

where X is the pixel value, W_i is the weight of the i^{th} Gaussian, m is the number of Gaussians, and η is the Gaussian form characterized by the mean μ_i and covariance Σ_i. Assuming R (Red), G (Green), B (Blue), and T (Temperature) channels are independent, each pixel is represented by its first order statistics for each respective channel as follow:

$$\mathbf{I_R = <Fitness_R, W_{R_1}, \mu_{R_1}, \sigma_{R_1},\ldots,W_{R_m}, \mu_{R_m}, \sigma_{R_m}>,}$$
$$\mathbf{I_G = <Fitness_G, W_{G_1}, \mu_{G_1}, \sigma_{G_1}, \ldots, W_{G_m}, \mu_{G_m}, \sigma_{G_m}>,}$$
$$\mathbf{I_B = <Fitness_B, W_{B_1}, \mu_{B_1}, \sigma_{B_1},\ldots,W_{B_m}, \mu_{B_m}, \sigma_{B_m}>,}$$
$$\mathbf{I_T = <Fitness_T, W_{T_1}, \mu_{T_1}, \sigma_{T_1},\ldots, W_{T_m}, \mu_{T_m}, \sigma_{T_m}>}$$

where Fitness is an evaluation value assigned to the mixture model for a given channel (see Section 3.3). Therefore, background model for a pixel is represented by concatenating the representations of all the channels, which represents a solution instance. An evolutionary-based search algorithm (see Section 3.3) is used to search the solution space for an optimal background representation.

3.2 Physical Models

The algorithm shown in Table 1 uses the physics-based predictions in its evaluation phase. Models of bi-directional reflectance distribution functions (BRDF) and thermal equilibrium based on conservation of energy are used to predict surface color and temperature in the visible and longwave IR. The models are briefly described here.

3.2.1 Physical Models of Reflectance

Several reflectance models including the Lambertian, Phong, dichromatic [17] and Ward [22] models have been developed to describe the reflectance due to normal, forescatter and backscatter distributions. We utilize the dichromatic model:

$$L(\lambda, \hat{e}) = L_i(\lambda, \hat{e}) + L_b(\lambda,\hat{e}) = m_i(\hat{e})\, c_i(\lambda) + m_b(\hat{e})\, c_b(\lambda);$$

where L is the total reflected intensity, L_i and L_b are reflected intensities due to surface and subsurface respectively, m_i and m_b are geometric terms, c_i and c_b are relative spectral power distribution (SPD) of the surface and subsurface respectively, and ê is a vector representing incident and reflected light angles with respect to the surface normal. The dichromatic model is useful in describing the reflection from inhomogeneous opaque dielectric materials (e.g., plastics). It is also useful in describing material colors since the SPD of the reflected light due to subsurface is decoupled from the geometric terms. To calculate the invariant body color, the image is segmented into regions with uniform reflectivity. For each region, pixel values in the RGB space are formed into a matrix M of size n × 3 where n is the number of rows (pixels) and 3 represents R, G, and B values. Singular value decomposition is then applied to M and the singular vector corresponding to the largest singular value is selected as the body color (c_b), which is the predicted surface color [11].

3.2.2 Thermal Physical Model

For predicting surface temperatures in the longwave IR, the following conservation of energy model is used. $E_{in} = E_{out}$; $E_{out} = E_{rad} + E_{cv} + E_{cd}$; Where E_{in} is the input energy, E_{out} is the output energy described by three phenomenon E_{rad} (energy radiated), E_{cv} (energy convected), and E_{cd} (energy conducted). Models for each energy flux is described in details in [11]. Briefly the following models are used to describe each of the above fluxes:

$$\mathbf{E_{in}} = E_{direct} + E_{skylight} + E_{atm}$$

$$\mathbf{E_{direct}} = (1089.5/ma)\, e^{(-0.2819\ m_a)}$$

$$\mathbf{E_{atm}} = E(BB,Ta)\ \{1\text{-}[0.261\ e^{-7.77 * 10\text{-}4\ (273\text{-}Ta)2}]\}\ \text{where}$$

E_{direct} = direct irradiation due to sun, $E_{skylight}$ = irradiation due to sky ≈ (40-70 W/m^2), E_{atm} = irradiation due to upper atmosphere, m_a = The number of air masses (m_a ≈ secant(Z)), T_a = Air temperature, E(BB, T_a) = radiation of a blackbody at T_a temp, and Z = sun's Zenith angle.

E_{rad} is estimated based on Stephen-Boltzman law:

$E_b = \sigma T^4$, where $\sigma = 5.669 \times 10^{-8}$ watts/m^2 Kelvin^4 and the subscript *b* is for blackbody which is capable of 100% absorption (or emission) of energy.

The convected heat flux is given by:

$$\mathbf{E}_{\mathbf{cv}} = \mathbf{h}_{\mathbf{cv}} (\mathrm{Ts} - \mathrm{T}\infty)$$

where h_{cv} is the convective heat transfer function which is a complex phenomena, Ts and T∞ are surface and fluid temperatures respectively. For laminar flow, h_{cv} can be roughly estimated by the following empirical model:

$$\mathbf{h}_{\mathbf{cv}} = 1.7 \;\; |\mathrm{Ts} - \mathrm{Ta}|^{1/3} + (6\ \mathrm{Va}^{0.8}) / \mathrm{L}^{0.2}$$

where Va = wind speed; L = characteristic Lateral dimension of surface, Ts and Ta are surface and air temperature respectively.

The conducted heat flux is described by:

$$\mathbf{E}_{\mathbf{cd}} = \mathrm{A}\ (\mathrm{T2} - \mathrm{T1}) / (\mathrm{L} / \mathrm{k})$$

where A is the area, T2-T1 is the differential temperature and L/k is called the thermal resistance or R-value and is tabulated for many materials. The above equilibrium model is solved for Ts, which is the predicted temperature.

2.3.3 Background Model Estimation

As mentioned in Section 3.1, a pixel is represented by concatenating mixture of Gaussian models of all its channels R, G, B, and T. In the mixture model, a single Gaussian is parameterized by W, μ, and σ; therefore, finding the best representation for a pixel with 4 channels represented by m Gaussians in each channel, requires searching in a $4 \times 3 \times m = 12m$ dimensional space. There are several search algorithms including brute force (e.g. depth first, breadth first), gradient methods (e.g. neural networks), heuristic methods (e.g., best first, beam search, A*), and genetic algorithms (GA).

Brute force methods are computationally expensive. Gradient-based techniques are suboptimal and may converge to local maxima. And, heuristic methods suffer from the curse of dimensionality. Genetic algorithms search from a population of individuals, which makes them ideal for parallel architectures. They have the potential to provide the global maximum.

Genetic algorithms are based on evolutionary process, examples of which are abundant in nature. In a typical GA the solution to a problem is encoded in each individual representation. A population of these individuals is randomly created. This population represents the location of individuals in the search space. An evaluation function (fitness function) that plays the role of the environment, rating individuals in terms of their fitness, is defined. The fitness function is used to rank individuals in the population. To continue exploring the search space, new populations are generated where individuals in the new population are selected based on the performance of their predecessors. In other words, solutions that have higher fitness value (e.g., better representations) are given more chance of being propagated in the next generation. In order to explore this search space more effectively, randomization is introduced in the selection of the individuals. There are two main operators for this randomization, referred to as crossover and mutation. Crossover is an operation where two individuals swap portions of their representation in random, effectively creating new offspring (solutions) encoding part of their parents (old solutions). Mutation is an operator that randomly, usually with low probability, changes a representation, for example, flipping a bit in a bit string. By applying the crossover, mutation and selection operators, the GA effectively explores the search space in a parallel fashion.

The Cooperative Coevolution (CC) algorithm utilized here is a recent evolutionary, GA-like, algorithm [14]. Like the GA algorithm, the CC algorithm explores the solution space in a random fashion. As in GA, the CC algorithm applies the operators crossover, mutation, and selection to generate potential solutions. However, in CC, the representation of a solution is broken down into sub-parts, each of which encodes part of the solution and is evolved separately. Therefore, sub-populations are generated and maintained in each generation of the CC algorithm. In this manner, the opportunities for searching and exploring different solution subspaces are increased. By comparing the algorithms in Figure 2, it is clear that the major difference between these two models lies in how the evaluation of individuals is performed. As stated earlier the evaluation in the GA model is performed on an individual (as a whole) in a population; on the other hand, in the CC model, individuals from separate sub-populations must come together to create an "organism" that is viewed as the solution. Hence, in the CC model, an individual cannot provide a meaningful solution to the problem and requires the cooperation of individuals from other sub-populations.

```
Procedure GA( )
initialize population
loop
  evaluate individuals
  store best individual

   select mating candidates
   recombine parents and use their
  offspring as the next generation
until stopping condition

return best individual
```

```
Procedure CC( )
initialize subpopulations
loop
evaluate organisms (solutions)
store best organism
for each subpopulation
  select mating candidates
  recombine parents and use their
   offspring as the next generation
end for
until stopping condition
return best organism
```

Figure 2: Comparing a typical GA and CC algorithm.

The success of CC depends on 4 criteria:

1) Problem decomposition,

2) Interdependability,

3) Credit assignment, and

4) Population diversity.

Our sensor fusion algorithm satisfies all four criteria's since

a) our problem is naturally decomposed (color video and IR),

b) our representation (mixture of Gaussians for all the 4 channels (R, G, B, and T)) provides interdependencies between subcomponents,

c) the objective or fitness function minimizes the discrepancy between the physics-based prediction and the actual observations in both IR and video, and

d) population diversity is maintained by roulette wheel selection method.

As mentioned, an important part of the evolutionary algorithm is the evaluation function, referred to as the fitness function. We provide a suitable fitness function that integrates the statistics collected by the system and the physical models that are directed by the contextual information (environmental conditions). The cooperative coevolutionary (CC) algorithm is used to select an optimal representation for a pixel background based on the recent past observations, classification (background vs. foreground) results, and physics-based predictions.

Fitness Function

For each channel, a population of individuals (see Section 3.1) is initially created randomly. These individuals are maintained for both the video channels (R, G, and B) and the thermal channel (T).

Briefly, the CC algorithm (see Table 1 and Figure 3), works as follows: Initially, 4 groups of individuals of type I_R, I_G, I_B, and I_T are randomly initialized. Each group is called a sub-population and each member of a sub-population is referred to as an individual, which is also assigned a fitness value. The fitness value (see Figure 3) is a measure of goodness and indicates how well that individual represents the background for its respective channel.

Individuals are rewarded when they perform well together as a team and punished when they perform poorly. This is the key concept in cooperative coevolutionary paradigm. In the example in Figure 3, to evaluate an individual in the red channel, it is combined with representatives from other sub-populations (in this case the representative of a sub-population is an individual with the highest fitness value); an organism is then created. The fitness value of the organism, $F_{organism}$, indicates how well the individual (I_R in this example) fits with other channels. In another words, the fitness value indicates the contribution of this individual as part of a whole solution.

The evaluation of the fitness function requires a training set. This training set is a recent past history, which includes observations, predictions, and classifications for each pixel and is kept in a QUEUE. To initialize the algorithm, initial n frames of background from all channels R, G, B, and T are collected and kept in a memory queue. Similarly a physics-based prediction for each pixel for each frame is kept in the memory queue. Since the initial n frames are assumed to be background, the groundtruth at the initialization stage is known (e.g., all pixels represent background). As a new frame is observed and pixels classified as either background or foreground, this training set is updated in a LIFO (Last In First Out) manner.

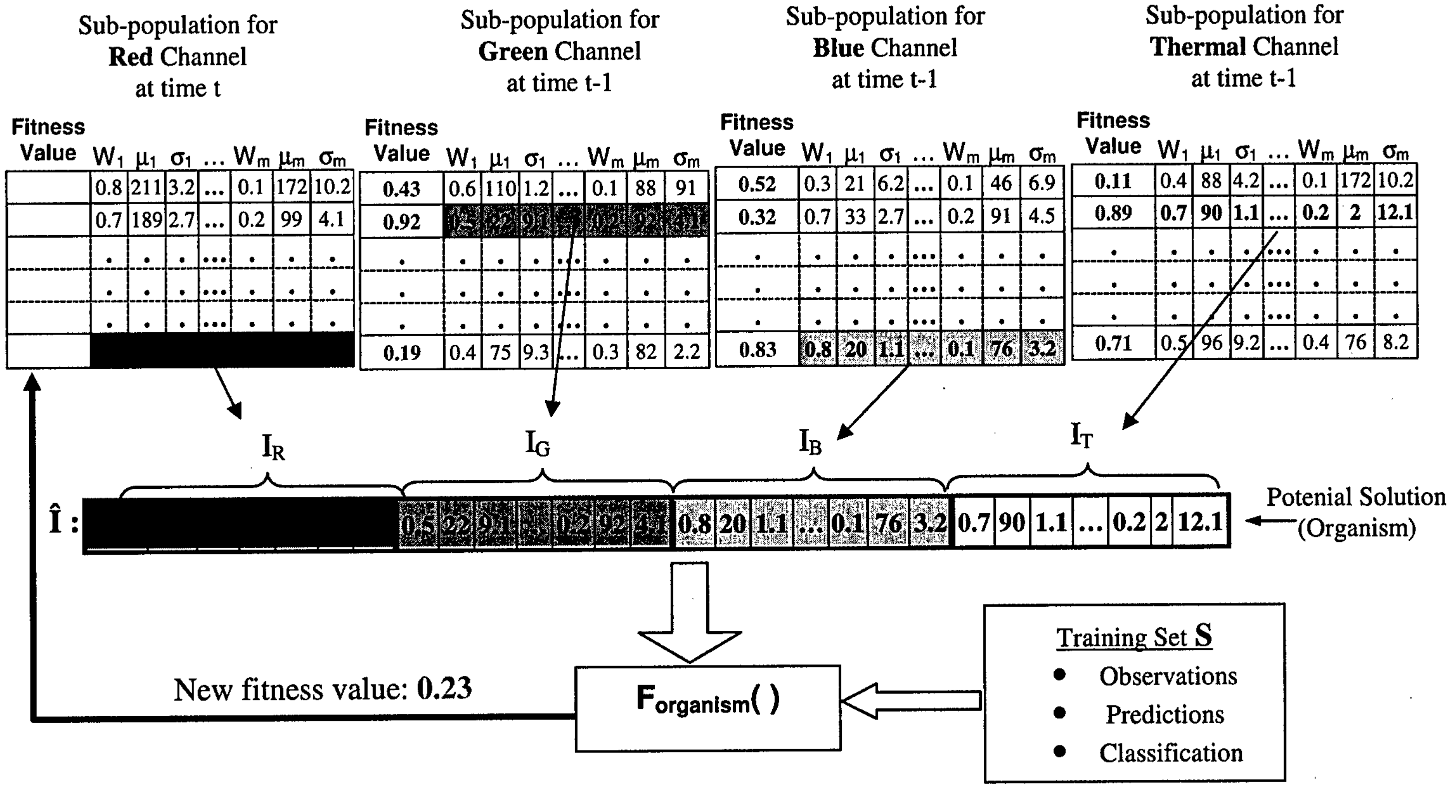
Pixel classification is the result of detection where a pixel is classified as background if its value falls within 3σ of any of its Gaussians for all the channels; else, it is considered a foreground.

For each channel, let an individual I_Y in a population be represented as in Section 3.1 where Y represents a channel, $Y \in \{R, G, B, T\}$. Let:

Y_{Xob_j} = Observed value of a pixel X at j^{th} frame for channel Y, j = 1 .. n ; n = size of the window in the past.

Y_{Xp_j} = Predicted value of a pixel X by physics for the j^{th} frame for channel Y.

$P(Y_X)$ = The probability distribution function for the pixel X for channel Y .



Sub-population for **Red** Channel at time t

Fitness Value	W_1	μ_1	σ_1	...	W_m	μ_m	σ_m
	0.8	211	3.2	...	0.1	172	10.2
	0.7	189	2.7	...	0.2	99	4.1
	•	•	•	•••	•	•	•
	•	•	•	•••	•	•	•
	•	•	•	•••	•	•	•

Sub-population for **Green** Channel at time t-1

Fitness Value	W_1	μ_1	σ_1	...	W_m	μ_m	σ_m
0.43	0.6	110	1.2	...	0.1	88	91
0.92	0.5	22	91	...	0.2	92	4.1
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
0.19	0.4	75	9.3	...	0.3	82	2.2

Sub-population for **Blue** Channel at time t-1

Fitness Value	W_1	μ_1	σ_1	...	W_m	μ_m	σ_m
0.52	0.3	21	6.2	...	0.1	46	6.9
0.32	0.7	33	2.7	...	0.2	91	4.5
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
0.83	**0.8**	**20**	**1.1**	**...**	**0.1**	**76**	**3.2**

Sub-population for **Thermal** Channel at time t-1

Fitness Value	W_1	μ_1	σ_1	...	W_m	μ_m	σ_m
0.11	0.4	88	4.2	...	0.1	172	10.2
0.89	**0.7**	**90**	**1.1**	**...**	**0.2**	**2**	**12.1**
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
•	•	•	•	•••	•	•	•
0.71	0.5	96	9.2	...	0.4	76	8.2

Figure 3: Example of evaluating an individual in the red channel - Individuals with highest fitness value in their population from other channels at the previous generation are combined to form an organism (solution). The result is stored back for the individual in the red channel.

We keep a moving window of n previous frames for all the channels. This window serves as the groundtruth data, G, for training examples. Unlike most other works that only use the last or current observation (frame) to update the mixture of Gaussians, we elect to keep a window of frames. Let

$$\underset{\{j=1..n\}}{G_j} = \begin{cases} 1 & Background \\ 0 & Foreground \end{cases}$$

where {j = 1..n} represents the last n frames (e.g., G_1 = current frame, G_2 = previous frame, and so on), and G is used as part of the training set *S*. Initially G for the current frame is obtained by using the mixture of Gaussian parameters that are obtained at time t-1. After the learning process has been completed for the current frame, G for the current frame is updated based on the learned parameters of mixture of Gaussians. In order to relate statistics-based classification and the physics-based predictions, we introduce the following function, named *credibility function* for each channel:

$$C_Y = e^{-\alpha\left[\frac{1}{n}\sum_{j=1}^{n} G_j \frac{\left|Y_{X_{ob_j}} - Y_{X_{P_j}}\right|}{Y_{X_{ob_j}} + Y_{X_{P_j}}} + (1-G_j)\left(1-\frac{\left|Y_{X_{ob_j}} - Y_{X_{P_j}}\right|}{Y_{X_{ob_j}} + Y_{X_{P_j}}}\right)\right]}$$

where vectors G, Y_{Xob} and Y_{Xp} are defined as before and α controls the rate of decay of credibility function. As the observed values Y_{Xob} are closer to the predicted values Y_{Xp} for a particular classification G, then the value of the credibility approaches to1. For example, it is easy to verify that in the extreme case where a pixel is classified as the background pixel in all the previous n frames, and that the predicted pixel values matched the observed values, the credibility will be close to 1.

The physics-based prediction predicts color and thermal properties of the background, therefore, it will be more credible if the observed pixel value is classified as the background pixel, and the predicted pixel value agrees with the observed value. Similarly, if the physics predicts a very different value than observed value and the system has actually classified the pixel as the foreground, then the physics may still be credible. On the other hand, if the physics-based prediction is very close to that of the observed value but the system has classified the pixel as foreground, then the physics-based prediction may not be reliable and a low credibility must be assigned. This process is depicted in Table 2.

Table 2: Credibility table describing the relationship between the predicted and observed values.

	Difference of current observed with predicted by physics	
	High	**Low**
Classification: Background	LOW Credibility	HIGH Credibility
Foreground	HIGH Credibility	LOW Credibility

The statistical estimation of fitness function based on the recent past observations for an individual, in channel Y, is given by:

$$F(I_Y) = \frac{1}{n}\sum_{j=1}^{n}\left[G_j P(Yx_{ob_j}) + (1 - G_j)(1 - P(Yx_{ob_j})\right]$$

The above function is only based on the statistical properties of the current and past observations. Given $F(I_Y)$ and the credibility function C_Y for individuals for all channels R, G, B, and T, then, a fitness function for an organism (solution) made of both video and IR species (e.g., R, G, B, and T channels) can be realized as follows:

$$F_{organism}\ (<I_R, I_G, I_B, I_T>) = \frac{1}{4}\ [C_R\ F(I_R) + C_G\ F(I_G) + C_B\ F(I_B) + C_T\ F(I_T)]$$

The above equation is used for evaluating the organisms formed by the video and IR signals, in which the individual being evaluated, is part of a complete solution, see Figure 3.

The final solution is the organism obtained by selecting the best individual (e.g., individual with highest fitness value) from each subpopulation. This solution is used to classify the current pixel as background or foreground.

The parameter α adjusts the importance of the role the credibility function plays in the fitness function. α can be adjusted depending on how fast the credibility function is desired to be influenced by the agreement between the physics prediction and actual observations.

Figure 4 shows how the parameter α affects the rate of change in the credibility function. For the observed temperature of 285° K, if the predictions are credible but not as close to the observed values, then lower values of alpha are desired. On the other hand, if tight coupling between physics predictions and observations is required, higher values for α are desired.

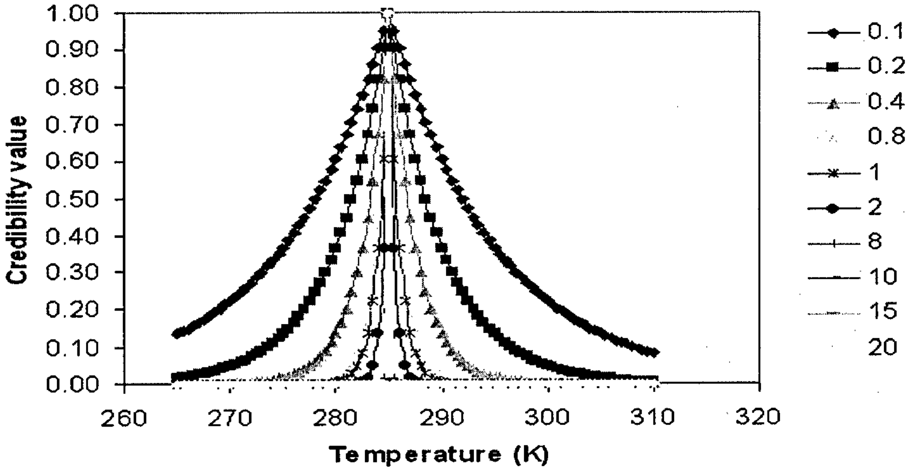


Figure 4: Credibility values for various values of alpha (α) for the thermal channel.

2.4. Experiments

The data was gathered at a typical urban location with the latitude 33° 50' 06" N and longitude 117° 54' 49" W, from 15:30:00 on January 21, 2003 till 14:24:00 January 22, 2003. Initially, from 15:30:00 till 17:07:04, data was collected at the rate of 1 frame every 2 seconds, then the temporal resolution was changed to approximately 1 frame per 10 second for the rest of the data collection period. Two cameras, a FLIR system thermal camera operating at 7-13 μm and an Intel web-cam operating in the visible range were utilized for data acquisition. The thermal camera was fully radiometric, which means that the pixel values obtained by the camera were thermal. The thermal camera included self-calibration that at specified intervals adjusted to internal thermal noise. The radiation-to-temperature conversion was done automatically by the camera for the default values of emissivity = 0.92, air and ambient temperatures = 280° Kelvin, distance to target = 100 m, and humidity = 50%.

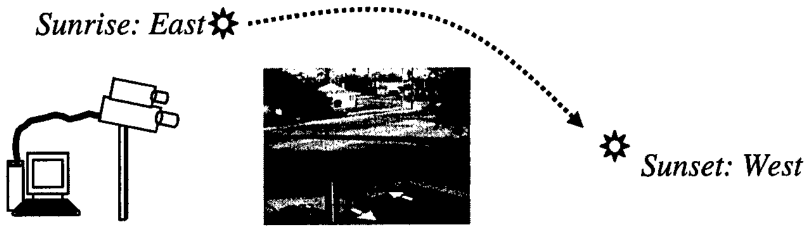


Figure 5. Position of the cameras with respect to the scene and the direction of the sun's path.

The video camera was attached on the top of the thermal camera on a tripod (see Figure 5). Both cameras were located 20 feet above the ground looking downward at the scene at an angle of approximately 25°. In addition to the thermal and the video cameras, a complete weather station was utilized to obtain weather data every minute. The weather station included an anemometer, humidity sensor, wind direction, two temperature sensors, and a barometer sensor. All sensors and the cameras were controlled by a PC. The data from the cameras and the weather station were synchronized through a software control.

To avoid temporal registration, both cameras were triggered simultaneously and in parallel. For spatial registration between the two cameras affine transformation was applied. For predicting correct reflectance and thermal predictions, a split and merge algorithm initially segmented the images for both cameras and a user initially labeled the segments into 5 regions, asphalt, concrete, grass, bush, and unknown. Only statistical properties were utilized for the unknown surface type.

2.4.1 Physical Model Estimation and Predictions

For surface color estimation, the dichromatic model was utilized. The results for the four different pre-segmented surfaces are given in terms of unit vectors in the RGB space. Due to lack of illumination during the nighttime, the values were obtained after sunrise and before sunset for various times and are given in Table 3 at an hourly illumination condition. The asphalt and concrete had similar vectors due to their neutral color attributes. On the other hand, the chlorophyll in the vegetation such as grass and bush causes the vectors to be shifted toward green. The higher variation in the reflectance of grass and bush are contributed by their surface specularity, which is not modeled by our algorithm.

Table 3. Surface body color estimation (c_b).

Time	Asphalt			Concrete		
	R	G	B	R	G	B
8:30	.5727	.5726	.5867	.5813	.582	.5687
9:30	.5714	.5716	.5889	.5791	.5797	.5732
10:30	.5773	.5714	.5862	.5824	.5824	.567
11:30	.5669	.5676	.5970	.5737	.5745	.5838
12:30	.5695	.5695	.5927	.5686	.5749	.5884
13:30	.5682	.5680	.5954	.5767	.5753	.5801
14:30	.5741	.5720	.5859	.5681	.5752	.5886
15:30	.5635	.5520	.6025	.5570	.5723	.6019
16:30	.5623	.5684	.6006	.5572	.5802	.594
17:30	.5544	.5668	.6095	.5566	.5813	.5935

Time	Grass			Bush		
	R	G	B	R	G	B
8:30	.6336	.7260	.2672	.5718	.6239	.5327
9:30	.6343	.7189	.2844	.5893	.6240	.5132
10:30	.6369	.7128	.2938	.5662	.6368	.5234
11:30	.6320	.7193	.2883	.5476	.6250	.5563
12:30	.6256	.7376	.2543	.5430	.6370	.5471
13:30	.6249	.7364	.2591	.5749	.6404	.5093
14:30	.6210	.7391	.2611	.5968	.6338	.4921
15:30	.6060	.7505	.2636	.5639	.6421	.5193
16:30	.6040	.7572	.2486	.6567	.6369	.4039
17:30	.6231	.7380	.2590	.6321	.6357	.4431

For surface temperature prediction, the thermal models of Section 3.2.2 were used. These predictions were used by the fitness function in section 3.3.1. We obtain the result of predictions superimposed on actual measurements by the thermal camera. The models were able to track temperature fluctuations for 4 different surface types. The average difference between the prediction and measurement for all surfaces were about 2°c with standard deviation of 1.87°c.

2.4.2 Detection Results

Moving object detection is performed after an initial background model is built. Once new thermal and video frames are available, they are registered. The registered image then contains red, green, blue, and temperature values at each pixel location. The cooperative coevolutionary algorithm is used to build the background model. Each pixel is updated independently. The background model is periodically updated to track the environmental changes. The following parameters were used in the cooperative coevolutionary algorithm to update the background models: number of species = 4; population size = 60; crossover = single point; crossover rate = 0.8; mutation rate = 0.01; maximum number of generations = 60; training data = 20 frames; number of Gaussians per sensor = 3; $\alpha = 0.5$.

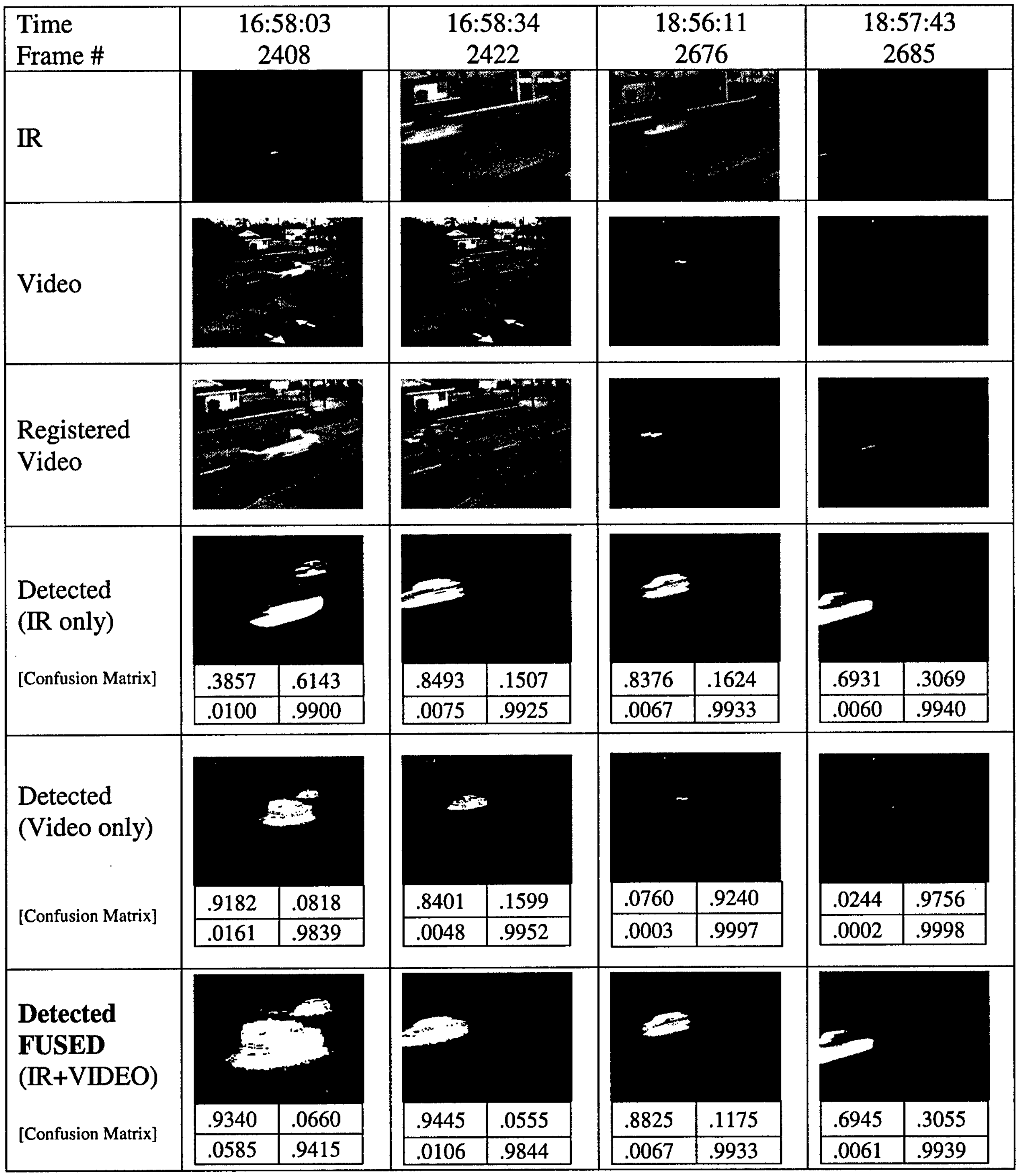
Once the background model is available, for each incoming frame, each pixel is compared to its corresponding model and if its value is within 3 standard deviation of any of its Gaussians, it is classified as a background pixel. This information is kept in a binary image where a detected moving pixel is a binary 1 (white) and a background pixel is 0 (black). These binary frames provide training data for the next background model update. In the following examples, in addition to the thermal IR and video frames, detection for each camera and the fused detection for the registered images are also provided. The following confusion matrix is given for the results for all the moving objects:

% Moving object correctly detected	% Moving object missed
% Background missed	% Background correctly detected

Figure 6. Performance measure.

- **Example 1:** Figure 7 shows example frames detected in the afternoon and early evening hours. During this period, illumination and heat exchanges are rapid. Depending on the heat stored and reradiated by an object and the background the object may be observed having very similar temperatures as the background (IR frames 2408 and 2685) or very different (IR frames 2422 and 2676). In frame 2408, video signal was much stronger, providing sharp contrast for the moving objects. Despite the lower performance of the IR, the objects were recovered by the video. Similarly, in frame 2422, the detection result of the IR was further enhanced by the registered video as is shown in the fused detected frame. Frames 2676 and 2685 are obtained during early evening hours. The video camera had a 25 lux minimum illumination requirement; therefore, although the scene was not totally dark, the video signal during the night time was very weak. This was compensated by the strong IR signal.

- **Example 2:** Figure 8 is an example where the detection algorithm relied heavily on one sensor, IR. Due to lack of illumination and video sensor's low sensitivity, objects could not be detected by video only. A good example is frame 2726 where a car and a person were in the scene. These were not observed in the video; however, they were present in the IR image and were clearly detected in both IR and the fused frame. Frames 2741, 6692 and 6718, indicate that the detection was not influenced by the video. The lights from the vehicles were visible and detected as part of the moving object, and the surface reflection from the lights did not affect the results.



Time Frame #	16:58:03 2408	16:58:34 2422	18:56:11 2676	18:57:43 2685
IR				
Video				
Registered Video				
Detected (IR only) [Confusion Matrix]	.3857 .6143 .0100 .9900	.8493 .1507 .0075 .9925	.8376 .1624 .0067 .9933	.6931 .3069 .0060 .9940
Detected (Video only) [Confusion Matrix]	.9182 .0818 .0161 .9839	.8401 .1599 .0048 .9952	.0760 .9240 .0003 .9997	.0244 .9756 .0002 .9998
Detected FUSED (IR+VIDEO) [Confusion Matrix]	.9340 .0660 .0585 .9415	.9445 .0555 .0106 .9844	.8825 .1175 .0067 .9933	.6945 .3055 .0061 .9939

Figure 7. Example 1: Mixed good and bad IR and video at various times in the afternoon and early evening.

This is due to the fact that the physics-based prediction assigns low credibility to the video signal; hence, low reflections are not detected as foreground. In effect this plays a role in deciding how important a camera's observations are. If a video pixel gets a low credibility, then its values are less meaningful; therefore, in order to observe a change, the signal must be strong (e.g., front head lights of a car). Since the front head lamps of most vehicles are halogen and radiate heat, they are also observed as part of the vehicle in the IR image, thus, they are also being detected as part of the vehicle.

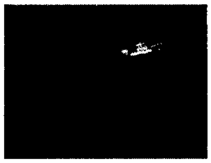
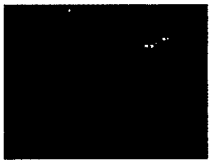
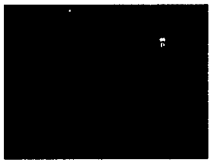
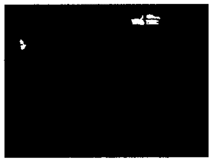
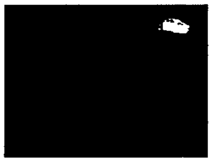
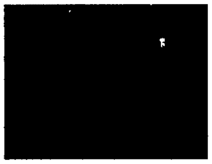
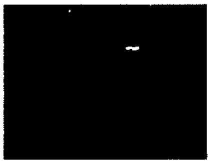
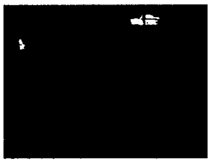
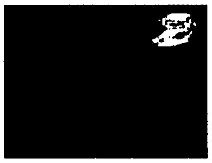
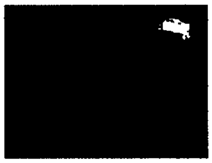
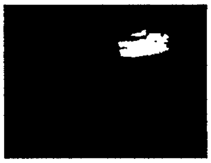
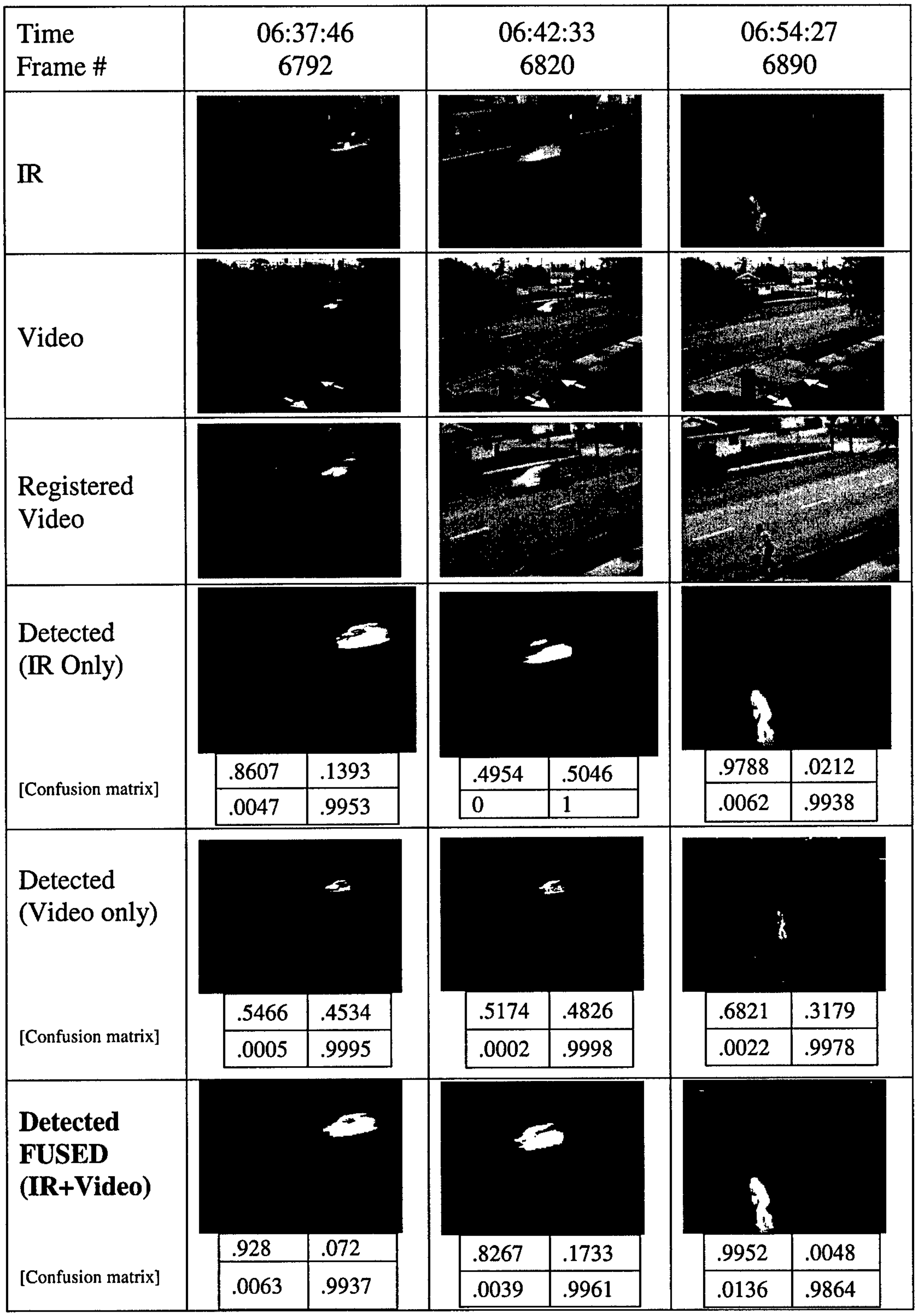
Time Frame #	19:04:42 2726	19:07:15 2741	06:20:43 6692	06:25:09 6718
IR				
Video				
Registered Video				
Detected (IR Only)				
Detected (Video only)				
Detected FUSED (IR+Video)				

Figure 8. Example 2. Good to excellent IR signal, bad video signal at night. (*Note: Due to lack of video contrast no groundtruth is obtained.*)

- **Example 3:** Figure 9 is an example of dramatic illumination changes during the early sunrise and early morning hours. During these periods, the environment changes radically due to the energy of the sun. The sensors must adapt to these rapid changes. Figure 6 shows the thermal changes on different surfaces that are tracked by the physics-based models. As shown, the slope of the temperature values changes radically during this period. However, the physics-based models are able to follow these changes and provide high credibility values that affect the background models built by the algorithm. As the illumination reaching the video camera is increased, the detection due to video gets better. This is shown in frames 6792 and 6820 where the video camera began participating in the detection process. This is indicated by the increase in the detection performance for the fused image versus the IR or video only images.

- **Example 4:** Figure 10 is an example of early morning, noon and early afternoon hours. As the sun comes up, the surfaces are heated up by the incoming energy from the sun, the increase in the surface temperatures approaches closer to the temperatures of some moving object surfaces. Depending on the moving object surface temperatures and emissivities, the contrast in the IR can be radically different. This is obvious between frames 6954 and 8646 for example. Frame 6954 represents an image in the morning with a person in the scene. Surface temperatures are still lower than that of the human body; moreover, human body's emissivity is high (0.98) compared to the background surfaces. The human is clearly visible in the IR image. Although not visible in the video image of frame 6954, the human is also in that image; this is clearer in the registered image. Both sensors provide good contrast in this case and the person is clearly detected.

Frames 8646 and 9350 show moving objects later in the day when surfaces have reached higher temperatures. In this case, it is possible to have a moving object that may have temperature close to the background surface as is indicated by both of these frames. On the other hand, video provide excellent signal and contrast. Many pixels are missing from the detected IR only, but the final fused detection recovers most of these missed pixels on moving objects.



Time Frame #	06:37:46 6792	06:42:33 6820	06:54:27 6890
IR			
Video			
Registered Video			
Detected (IR Only) [Confusion matrix]	.8607 .1393 .0047 .9953	.4954 .5046 0 1	.9788 .0212 .0062 .9938
Detected (Video only) [Confusion matrix]	.5466 .4534 .0005 .9995	.5174 .4826 .0002 .9998	.6821 .3179 .0022 .9978
Detected FUSED (IR+Video) [Confusion matrix]	.928 .072 .0063 .9937	.8267 .1733 .0039 .9961	.9952 .0048 .0136 .9864

Figure 9. Example 3: Fusion while illumination changes at sunrise.

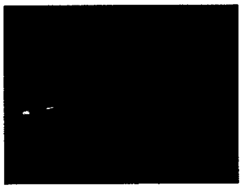
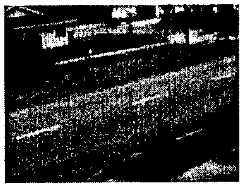
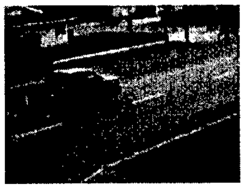
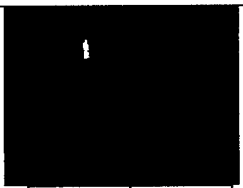
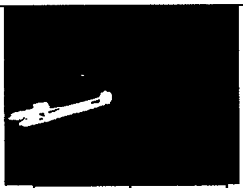
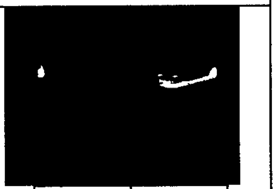
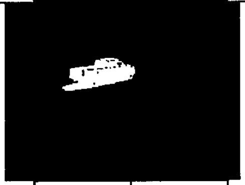
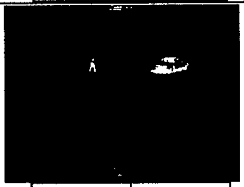
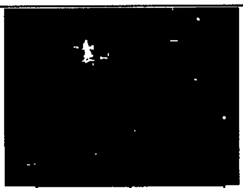
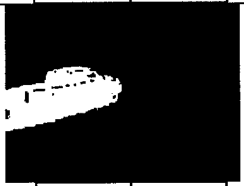
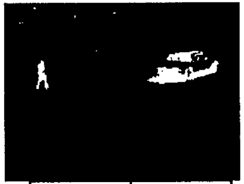
Time Frame #	07:05:20 6954	11:52:52 8646	13:52:29 9350
IR			
Video			
Registered Video			
Detected (IR Only) [Confusion matrix]	 .9106 .0894 .0001 .9999	 .2905 .7095 .0051 .9949	 .2379 .7621 .0007 .9993
Detected (Video only) [Confusion matrix]	 .9064 .0938 .0057 .9943	 .9333 .0667 .0054 .9946	 .5222 .4778 .0034 .9966
Detected FUSED (IR+Video) [Confusion matrix]	 .9343 .0657 .0039 .9961	 .9578 .0422 .0177 .9823	 .5636 .4364 .0049 .9951

Figure 10. Example 4: Mixed IR and good video signal.

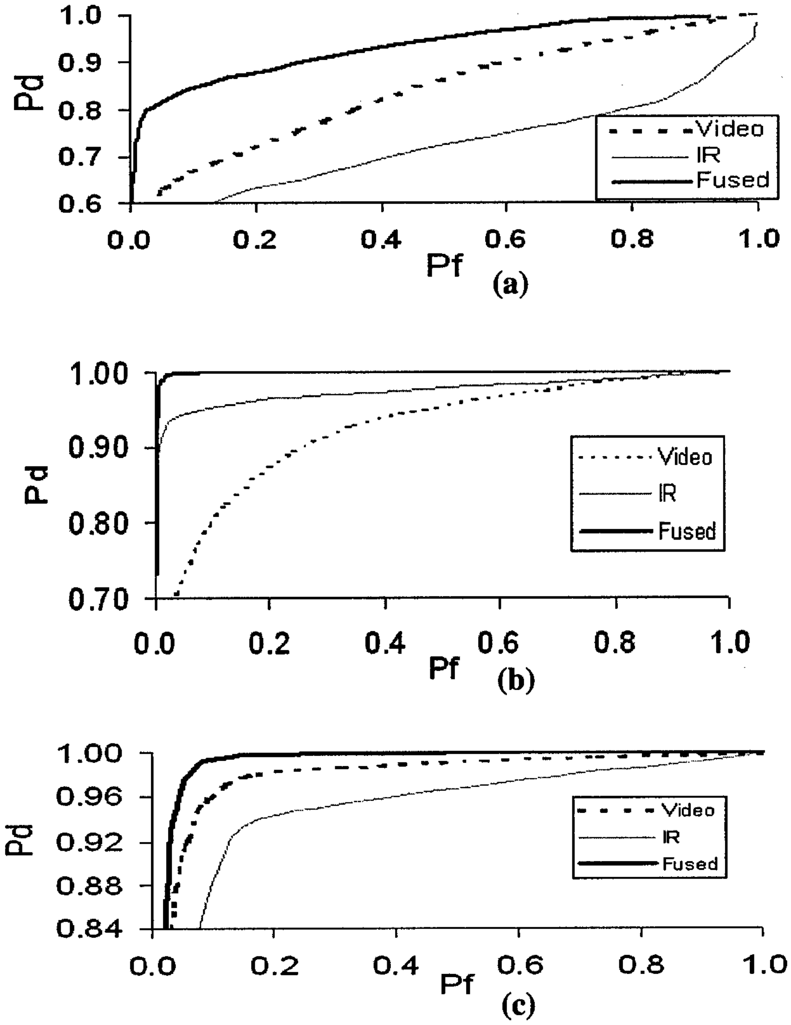
2.4.3 Performance Analysis

To compare the performance of the detection algorithm for sensor fusion, we utilize the Receiver

Operating Characteristic (ROC) curves and define the probability of detection as percentage of moving object pixels that are correctly detected and probability of false alarm as percent of background pixels that are classified as moving object. We selected frames representing afternoon, early morning and high noon for this analysis. The nighttime was not selected since no video signal was available at night (6:30 p.m. – 6:30 a.m.) and the detection algorithm relied only on the IR sensor; this was explained in example 2 above. The first ROC curve, Figure 11(a), represents an afternoon time. An example of this is frame 2408 in Figure 7. As is indicated by example 1 frame 2408 and this ROC curve, the video signal operated at a higher rate than the IR signal. The fusion method provides a higher level of performance than both the video and the IR.

The ROC curve of Figure 11(b) is an example of early morning hours. This figure is in contrast to that of Figure 11(a) in the afternoon. In this case the detection rates for both the IR and the fused image were high and the video sensor operated only nominally. This is again due to the fact that a great deal of energy has been dissipated to the environment throughout the night during early morning hours, and a large gradient may exist between natural surface temperatures and those of animated objects with internal sources of energy such as vehicles and humans. In addition, the video signal, as indicated in Figure 9, example 3, is rapidly changing due to the illumination changes when sun is rising in the sky.

The third ROC curve, Figure 11(c), is an example of how fusion can enhance the detection when both sensors may be operating at lower rates. This is an example when cooperation between sensors can play a complementary role. This is due to the fact that different sensors may detect different parts of an object. So, one expects sensor fusion to do much better in detecting more pixels on the object than any one of the sensors alone. This is observed from frames 8646 and 9340 of example 4 in Figure 10 when for example, the detected IR and video frames have detected different parts of the same object.



(a)

(b)

(c)

Figure 11. ROC curves for various periods of the day. (a) afternoon-evening, (b) early morning, (c) morning-noon.

These ROC curves also indicate that as the time of day changes, the dynamic sensor fusion introduced here can automatically adapt to environmental changes. This adaptation is also in the form of adapting to the best sensor at the time. The cooperation among sensors can also take on a complementary role when different cameras are able to detect different part of an object that may not be visible to the other. This adaptation is done continuously in a cooperative manner.

2.5. Conclusions

In this chapter a novel physics-based sensor fusion technique for moving object detection was introduced. The sensor fusion architecture integrated the statistical and phenomenology of the sensors in the visible and longwave IR through an evolutionary computational model. Our representation, mixture of Gaussians, along with the cooperative coevolutoionary search algorithm integrated the contextual information through the physics-based and statistical models. We showed that our fusion model adapted to various illumination conditions and is suitable for detection under variety of environmental conditions.

References:

[1] D. M. Buede, and P. Girardi, "A target identification comparison of Bayesian and Dempster-Shafer multisensor fusion," *IEEE Tran. on Systems Man & Cybernetics*, 27(5), pp. 569-577, Sept. 1997.

[2] M. Cristani, M. Bicego, and V. Murino, "Integrated region and pixel-based approach to background modeling," *IEEE Workshop on Motion and Video Computing*, pp. 3-8, Dec. 2002.

[3] J. Han and B. Bhanu, " Detecting moving humans using color and infrared video," *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 228-233, July 2003.

[4] I. Haritaoglu, D. Harwood, and L. Davis, "W4: real time surveillance of people and their activities," *IEEE Trans. Pattern Analysis and Machine Intelligence,* 22(8), 809-830, 2000.

[5] K. Heesung, Z.D. Sandor, and N.M. Nasrabadi, "Adaptive multisensor target detection using feature-based fusion," *Optical Engineering*, 41(1), pp. 69-80, Jan. 2002.

[6] T. Horprasert, D. Harwood, and L.S. Davis, "A statistical approach for real-time robust back-

ground subtraction and shadow detection," *Proc. FRAME-RATE Workshop held in conjunction with Intl. Conf. on Computer Vision*, pp. 1-19, 1999.

[7] R. Joshi, and A.C. Sanderson, " Minimal representation multisensor fusion using differential evolution," *IEEE Trans. on Systems, Man, & Cybernetics*, 29(1), pp. 63-76, Jan. 1999.

[8] A.J. Lipton, H. Fujiyoshi, and R.S. Patil, "Moving target classification and tracking from real-time video," *IEEE Workshop on Applications of Computer Vision*, pp. 8-14, 1998.

[9] B. Ma, and S. Lakshmanan, AO. Hero III., "Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion," *IEEE Trans. on Intelligent Transportation Systems*, 1(3), pp. 135-147, Sept. 2000.

[10] S. Majumbder, S. Scheding, and H.F. Durrant-Whyte, "Multisensor data fusion for underwater navigation," *Robotics & Autonomous Systems*, 35(2), pp. 97-108, May 2001.

[11] S. Nadimi and B. Bhanu, "Physics-based models of color and IR video for sensor fusion," *IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems*, pp. 161-166, July 2003.

[12] N. Nandhakumar, and J. K. Aggrawal, "Physics-based integration of multiple sensing modalities for scene interpretation," *Proc. of the IEEE*, Vol. 85, No.1, pp. 147-163, Jan. 1997.

[13] I. Pavlidis, et. al. "Automatic detection of vehicle passengers through near-infrared fusion," *Proc. IEEE Conference on Intelligent Transportation Systems*, pp. 304-309, Oct. 1999.

[14] M.A. Potter and K.A. DeJong, "A cooperative coevolutionary approach to function optimization," *Proc. of the 3rd Conference on Parallel Problem Solving from Nature*, pp. 249-257, 1994.

[15] D. Scribner, P. Warren, J. Schuler, M. Satyshur, and M. Kruer, "Infrared color vision: an approach to sensor fusion," *Optics and Photonics News*, pp. 27-32, Aug 1998.

[16] D. Scribner, P. Warren, and J. Schuler, "Extending color vision methods to bands beyond the visible," *Proc. IEEE Workshop on Computer Vision Beyond the Visible Spectrum: Methods and Applications*, pp.33-44, 1999.

[17] S. A. Shafer, "Using color to separate reflection components," *Color Research and Application*, 10(4), pp. 210-218, 1985.

[18] C. Stauffer and W.E.L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Analysis and Machine Intelligence* 22(8), pp 747-757, 2000.

[19] C.W. Therrien, J.W. Scrofani, and W.K. Krebs, "An adaptive technique for the enhanced fusion of low light visible with uncooled thermal infrared imagery," *Proc. Intl. Conference on Image Processing*, Vol. 3, pp. 405-409, 1997.

[20] M.E. Ulug, and C. McCullough, "Feature and data level fusion of infrared and visual images," *Proc. SPIE: Intl. Society for Optical Engineering*, 37(19), pp. 312-318, Apr. 1999.

[21] M.M.D. Viva, and C. Morrone, "Motion analysis by feature tracking," *Vision Research*, Vol. 38, pp. 3633-3653, 1998.

[22] G. Ward, "The RADIANCE lighting simulation and rendering system," *Computer Graphics (SIGGRAPH' 94)*, pp. 459-472, July 1994.

[23] A.M. Waxman, et. al. "Solid state color night vision: fusion of low-light visible and thermal infrared imagery," *MIT Lincoln Lab Journal*, 11(1), pp. 41-57, 1998.

[24] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Trans. Pattern Analysis and Machine Intelligence,* 19(7), pp. 780-785, 1997.

Chapter 3: An Evolutionary Agent Based Intrusion Detection System

Abstract: With the advent of more sophisticated network and computer systems, and participation of the public on the Internet, potential for abuse of the information and resources has greatly increased. This potential has flourished in variety of methods and techniques for obtaining access and/or manipulating a computer system and its contents. Whether it is software such as computer virus, worm, Trojan horse, etc. or an actual physical device such as network sniffers, that compromises the integrity of a computer system, the action is called intrusion. In this paper, a short description of intrusion detection is given and several examples of intrusion are provided. A formal definition of intrusion detection and various methods, techniques and intrusion detection systems (IDS) are described. The paper gives a formal description of an agent-based intrusion detection where evolutionary based techniques are explored. Moreover, a detailed architecture for a coevolutionary agent based system is given and the concept of a super agent is described in detail.

3.1. Introduction

Perhaps the best way to describe the concept of intrusion in computer and network systems is by investigating intrusions in real life and make comparisons. The following is a fictitious example: Bob owns a small grocery store with two employees. Before he locks up, he activates the alarm and then heads home. The next day, when Bob opens his store, he realizes that the alarm has been deactivated and some items are missing from his store. Did an intrusion occur? Should Bob call the police?

To answer these questions, one must know Bob's policy and whether the confidentiality, integrity and availability of his security system and his store been compromised. If none of the store employees have taken the items, then an intrusion has occurred. In addition, if they did not deactivate the alarm, then confidentiality of the alarm code has been compromised. Also, if there has been any damage to the alarm, the integrity and availability of the system has been compromised and an intrusion has occurred. However, if one of the store employees has deactivated the alarm, most likely no intrusion has occurred, this of course, depends on Bob's policy on whether the employees are allowed to deactivate the alarm and they can take any items with them or not!

This simple scenario shows us that in any organization, including a computer system, one must define the policies that affect the confidentiality, integrity and availability of the information and resources to the organization or the computer system. A *policy* is defined as the set of laws, rules, and practices that regulate how an organization manages, protects and distributes sensitive information [1]. To implement any security system, it is important to define a security policy for that system. A good and simple example of the security policy is the *file access policy* in a UNIX file system where permission for accessing a file is set by a user. A file access is defined as one of the read, write and execute actions and file access is limited to the user, his/her group, and everyone else. The file access policy is set by the user (owner) of the file. Any violation of the access policy is considered an intrusion. An intrusion can, therefore, be defined as any set of actions that attempt to compromise the integrity, confidentiality and availability of a resource [2], in this case the file system.

3.2 Intrusion Detection Techniques

In this section a summary of definitions, principles and techniques for intrusion detection are given. An intrusion detection system (IDS) can be categorized as a pro-active or a reactive system. Pro-active IDS systems are systems that actively, usually in real-time, enforce the security policy of the system. The enforcement is done at the time of any executed action. For example, if a user attempts to delete or alter a file he/she is not permitted to access to, he/she will be immediately notified of the illegal action. Other examples include accessing resources such as printers, disks or network ports that are not allowed, exceeding a process priority levels beyond what is allowed, or running or executing processes that are not allowed. Pro-active systems have a set of rules that define the policy of the system. These rules are either built into the system (for example, Unix File security policies) or provided as an external set, such as expert systems, where the rules can be modified, added or deleted to adapt to changes in the system.

A reactive system is a monitoring system that analyzes the user actions after they have been executed. In this sense, a reactive system is an offline process. A reactive system generally collects, analyzes and makes decision on whether an intrusion has occurred, after which the authorities are notified to take appropriate actions to stop the intrusion.

Both pro-active and reactive systems provide advantages and disadvantages of their own. Proactive systems are fast, execute in real time, and are easy to maintain and update the rules. Reactive systems have the advantage of discovering new intrusions (attacks) automatically, and utilize a great deal of available statistical information in the form of audit trail.

Intrusions are typically divided into 6 categories [3]:

1. Attempted break-ins, which are detected by atypical behavior profiles or violations of security policies,

2. Masquerade attacks, which are detected by atypical behavior profiles or violations of security,

3. Penetration of the security control system, which are detected by monitoring for specific patterns,
4. Leakage, which is detected by atypical use of system resources,
5. Denial of service, which is detected by atypical use of system resources,
6. Malicious use, which is detected by atypical behavior profiles, violations of security constraints, or use of special privileges.

The techniques to detect the type of intrusions mentioned above, fall into two general types, Misuse detection and Anomaly detection.

3.2.1 Misuse Detection

Methods that fall under the category of misuse detection directly encode the knowledge about the attacks. This encoding is in the form of patterns or well-defined rules that describe vulnerability of certain aspect of the system. For example, exploitation of the *finger* and *sendmail* bugs that are used in the Internet worm attack and their consequent expected behavior can be represented in a knowledge base. The knowledge base can be examined for appropriate bugs and their behaviors and appropriate action can be taken. There are 4 approaches to Misuse detection, 1) Expert Systems, 2) Key Stroke Monitoring, 3) Model-based Intrusion Detection, and 4) State Transition Analysis.

3.2.1 Expert Systems: These are computing systems that represent and reason about knowledge in a certain domain. The attacks are usually coded as *if-then* rules in the knowledge base where certain conditions in the *if* part of the rule must be satisfied before an action in the *then* part is triggered. In another words, expert systems are modeled in such a way as to separate the rule-matching phase from the action phase. In this way, the control reasoning is separated from the formulation of the problem solution. Its use is mainly deduction of the occurrence of an intrusion based on the knowledge or available data. The main advantages of the expert systems are the separation of the rules from the actions and easy upgradeability of the rules as new intrusions are discovered. Chief among the main disadvantages of expert systems is the knowledge base itself, which can only be as good as the expert that designs it. If an expert misses encoding certain intrusions, those intrusions can never be detected; moreover, other issues such as maintaining the knowledge base and the quality of the rules become major software engineering problems.

3.2.2 Model Based Systems: They combine models [4] of misuse with evidential reasoning to support conclusions about its occurrence. This approach states that certain scenarios are inferred by certain other observable activities. If these activities are monitored, it is possible to find intrusion attempts by looking at activities that infer a certain intrusion scenario. The model based intrusion detection has three important modules [4]:

a) The anticipator – the purpose of this module is to try to predict the next step in the scenario that is expected to occur. This module uses active and scenario models. A scenario model is a knowledge base with specifications of intrusion scenarios.

b) The planner – the purpose of the planner is to translate the hypothesis formed by the anticipator into a format that shows the behavior, as it would occur in the audit trail. It uses the predicted information to plan what to search for next.

c) The interpreter – the purpose of the interpreter is to search for the data generated by the planner in the audit trail.

The system continually accumulates audit data and evidence until a threshold has been met; at which point, a signal indicates that an intrusion has occurred. Because the planner and the interpreter know what they are searching for at each step, the large amounts of noise present in audit data can be filtered, leading to performance improvements. Since the system can predict the attacker's next move based on its model, these predictions can be used to verify an intrusion hypothesis, to take preventive measures, or to determine what data to look for. Despite these advantages, some of the major disadvantages of such systems are: 1) patterns of intrusions must be identified and recognized, 2) if a certain scenario/behavior is being tested, the pattern associated with that scenario must occur, and 3) intrusion pattern must be easily distinguishable from those of the normal patterns.

3.2.3 State Transition Analysis: These technique represents the monitored system as a state transition [5]. A system state represents a state in the attack pattern. The states are connected through transitions that occur according to Boolean assertions at the state node. An intrusion occurs when the system goes from a safe state to an unsafe state. In this manner, a state transition diagram represents safe vs. intrusive activities/behaviors. An advantage of this model is that it can detect attacks that can span across multiple user sessions. A disadvantage of this technique is that it cannot detect certain attacks such as denial of service attacks, failed logins, and passive listening because they are either not recorded by the audit trail or cannot be represented by the state transition diagram. Furthermore, attack patterns can specify only a sequence of events, rather than more complex forms.

3.2.4 Keystroke Monitoring: As its name implies, this techqnique monitors and searches for certain keystroke patterns. The advantage of this technique is its simplicity; unfortunately there are a myriad ways of implementing intrusions using different keystrokes. In addition, certain features in some Unix shells such as aliases also defeat this technique.

3.3 Anomaly Detection

The techniques falling under the anomaly detection types of intrusion detection systems assume that activities due to intrusions are different from that of system's normal behavior. If a profile of the normal system behavior can be established, it should at least be theoretically possible to find abnormalities that manifest themselves as deviations from the normal behavior. Some of the techniques for anomaly detection are:

1. **Statistical based**: In these approaches [6, 7, 16] a set of measures are derived to for example profile behavior of subjects. If there exists a sufficient variance in the present profile from that of original profile, an intrusion is likely occurred. One of the advantages of this technique is that it can potentially actively learn the behavior of users and it is more sensitive than the human experts.

2. **Predictive pattern generation**: This method [8, 9] of intrusion detection tries to predict future events based on the events that have already occurred. This is usually formed by a rule based sequential patterns. In this approach, rules for intrusion patterns are given as events with probabilities. For example a rule may specify that if a certain number of events $E_1..E_n$ have occurred then a number of possible events E_{n+1} .. E_k each with a different probability could occur. This approach can detect anomalous activities that are difficult with traditional

methods and is highly adaptive to changes; however, if the intrusion scenarios are not described by the rules, they will not be flagged intrusive.

3. Neural Networks: A neural network [4, 10] is trained on typical sets of commands entered by a user. Once the network is trained, it is presented with user command profiles already captured. The neural network is then used to detect deviations in the user profile from that of the trained scenario. Some of the advantages of this approach are that it copes well with noisy data, its success does not depend on any statistical assumptions about the nature of the underlying data, and it can be modified easily with the new user community. The disadvantages are that the window size for training (or testing) may result in false positives or false negatives and the network topology is determined after many testing and trials.

Since all anomaly detection methods rely on profiling and learning normal behaviors in order to distinguish deviation from it, almost all outlier detection schemes such as nearest neighbor approaches [11, 12, 13, 14], density based techniques [15], and clustering based techniques [16] are viable approaches.

3.4 Autonomous Agents for Intrusion Detection

One interesting approach that has been pursued by researchers at Purdue University is the idea of a distributed IDS system (vs. a single large monolithic IDS) [17]. They introduce the concept of autonomous agents for IDS where an agent is defined as a system that tries to fulfill a set of goals in a complex dynamic environment. In the context of an IDS system, the agent is a detector for anomalous behavior in a computer system; therefore, an agent is an IDS system itself. Advantages of such an approach are many including efficiency, fault tolerance, resilience to degradation, extensibility, and scalability. The agents are software programs that are automatically generated and maintained by the system. Since the agents are autonomous, many of them can exist simultaneously. This results in fault tolerance where if one or more agents are for one reason or another disrupted, others continue to monitor for intrusions. Moreover, agents can potentially be specialized to audit certain aspect of the system. For example, an agent can be trained for Input/Output activities, another for Network Activities and yet another for process resource allocation activities and so on. In [17] a simple agent-based model is provided for intrusion detection and intrusion detection is viewed as anomaly detection. Since the idea of the approach described in this report has the same spirit with the agent based IDS described in [17], this system is explained in more details.

The system proposed in [17] is an agent-based system where an agent is a lightweight program that observes only one aspect of the overall system. Agents can be developed to inspect a particular aspect of the system. For example, an agent can monitor the network, another inspect the CPU utilization, another for I/O, etc. The architecture of the agent based IDS is depicted in Figure 1.

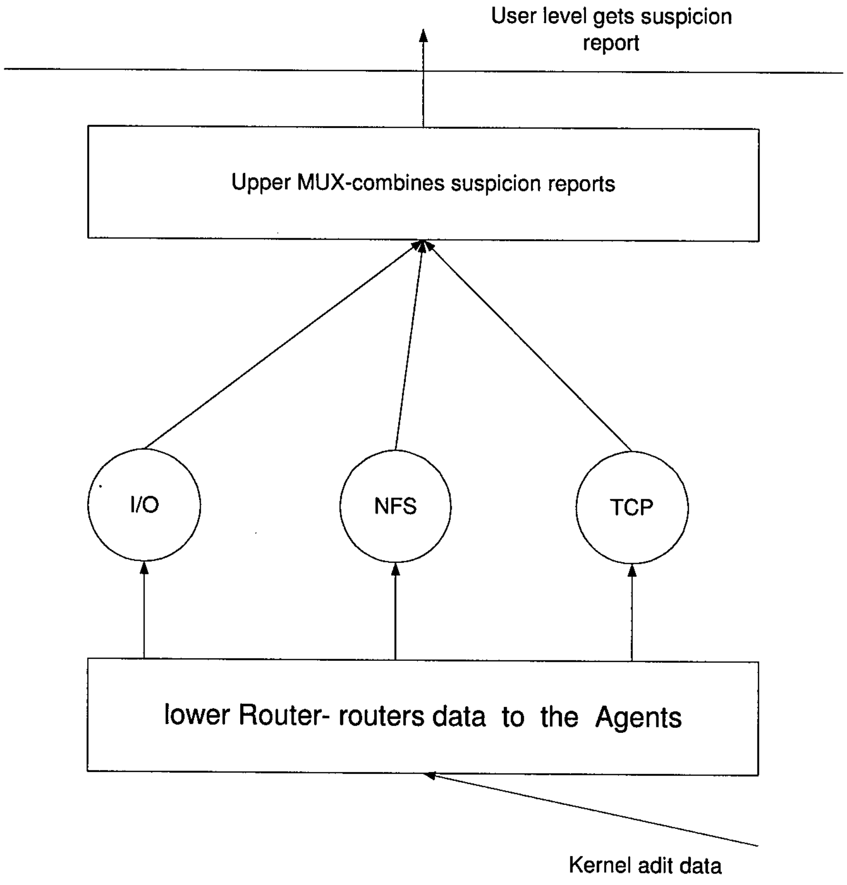

User level gets suspicion report
Upper MUX-combines suspicion reports
I/O
NFS
TCP
lower Router- routers data to the Agents
Kernel adit data

Figure 1. Overall Agent-based IDS architecture proposed in [17].

The input to the system is audit data. This data can come from a variety of places for example, web log files, process tables, TCP/IP, kernel memory traces, user profiles, and so on. The raw audit data is first scanned and usually parsed and decoded for input to the agent code. Each agent performs its computation and then reports a suspicion value. The suspicion values of all agents are gathered (multiplexed) and a final suspicion report is generated. This report can then be evaluated by the system administrator or security officer. The core of this architecture relies on the generation of the agent code. Note that Figure 1 does not specify how the agent codes (indicated by circles, I/O, NFS, TCP) are designed. One obvious approach is to replace these agent codes with that of a human expert. The advantage of human-based agent design is the efficiency in the code design, and the bias and expertise that the human designer introduces in the agent code. On the other hand, although the code design and actually implementation might be efficient, implementing changes, for example, due to new modes of attack might be cumbersome and error prone by the human designer. In addition, the expert's bias in designing the agent then fixes the behavior of the agent, the agent will be as good as its designer. Moreover, designing agent code is not a trivial task and may require enoromous amount of testing and interaction with the computing system before a useful agent is designed and trained.

The agent design can be automated. The automation requires a set of metrics that can abstract the system operation. The metrics can be computed from a set of parameters such as CPU utilization, number of TCP packets, IP addresses, number of connections, and so on. These parameters must encapsulate important and relevant information that can be useful for detecting anomalous behaviors.

An agent can be viewed as a program or process that utilizes metrics to detect anomalous behavior. Once the metrics are defined, and the training scenarios are available, agent codes can be automatically constructed and verified. One such method of constructing agents capable of learning is through evolutionary methods such as genetic algorithms, or genetic programming. In the following sections, several metrics are introduced. These metrics are utilized to efficiently construct agent codes. Moreover, the idea of a constructing a *super agent* through coevolution and its efficacy is described.

3.4.1 Genetic Programming and Coevolution: One can view an agent (a program code) as a solution to a search problem. The search problem is to find a good (hopefully the best) program among all possible programs that can detect an attack. A computer code can be viewed as a point entity in a multidimensional domain, where each dimension is a primitive describing part of the agent's language.

Example: Assume our language is very limited and contains only conditional statement *if-then*, conditional operators (<, >, and =), logical operators (*and*, *or*), and operator *action()*. Although these primitives might be too limited to be expressive for a computer language, most production systems require very limited number of primitives such as the ones given above. The following codes are examples of statements in our hypothetical language:

(a) If X >2000 then action(STOP_CONNECTION);

(b) If X<2000 and X>100 then action(ALERT_USER);

(c) If Y=1493 or Y=197 then action(CLOSE_PORT);

If X represents number of TCP connections and Y represents port numbers, then the first two statements are programs that check for number of TCP connections and the third statement may be a program that checks for access to a particular port.

Generally, programs are a set of statements that can be represented as multiple parse trees. A parse tree is a tree with nodes representing operators/primitives and each parse tree can represent a statement as the example given above. A list of parse trees (statements) represents a program; therefore, a program can be as small as a single parse tree or multiple parse trees. For the example above the three statements can be represented by the parse trees given in Figure 2.

Finding such parse trees that can represent useful rules for detecting anomalous behaviors or IDS, is a hard problem. This is mainly due to the unlimited number of possible combinations of codes that can be generated. An efficient search paradigm that is useful for such huge search spaces is based on evolutionary processes such as genetic algorithms (GA) and genetic programming (GP).

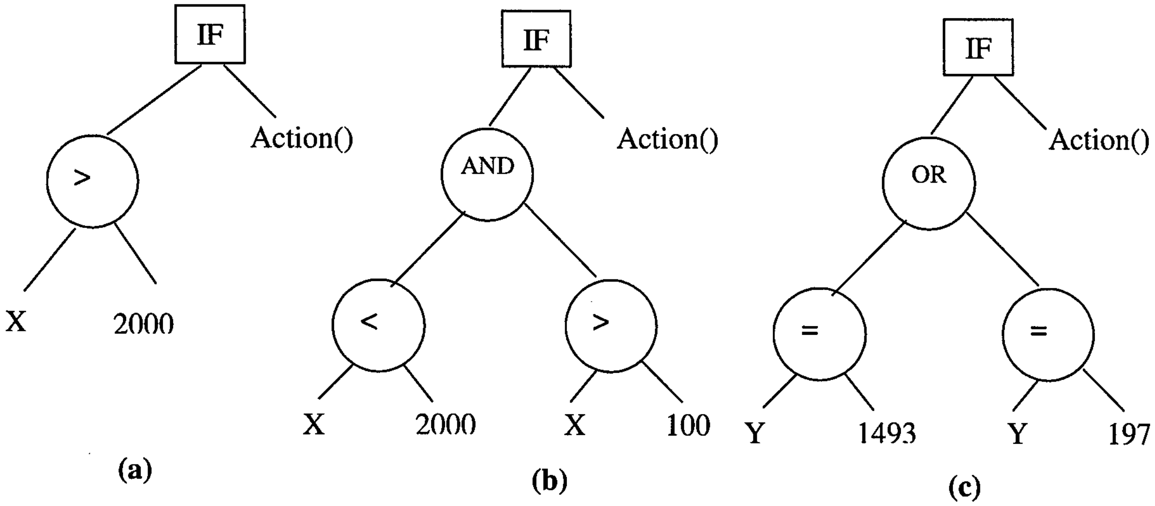


Figure 2. Parse trees corresponding to the example (a), (b) and (c) respectively.

In a typical GA the solution to a problem is encoded in each individual representation. A population of these individuals is randomly created. This population represents the location of individuals in the search space. An evaluation function (fitness function) that plays the role of the environment, rating individuals in terms of their fitness, is defined. The fitness function is used to rank individuals in the population. To continue exploring the search space, new populations are generated where individuals in the new population are selected based on the performance of their predecessors. In other words, solutions that have higher fitness value (e.g., better representations) are given more chance of being propagated in the next generation. In order to explore this search space more effectively, randomization is introduced in the selection of the individuals. There are two main operators for this randomization, referred to as crossover and mutation. Crossover is an operation where two individuals swap portions of their representation in random, effectively creating new offspring (solutions) encoding part of their parents (old solutions). Mutation is an operator that randomly, usually with low probability, changes a representation, for example, flipping a bit in a bit string. By applying the crossover, mutation and selection operators, the GA effectively explores the search space in a parallel fashion.

The operation in GP is identical to that of GA, that is operations crossover, mutation, and selection are performed similarly. The main difference between a GP and GA is that in the GA the solution (or search space) is encoded as bit strings whereas in GP the solution is encoded as programs, which are codes normally represented as parse trees[1]. The genetic operators are then modified to swap portions of parse tree (subtrees) for crossover, and changing a node for mutation.

Another recent evolutionary, GA-like, algorithm is the Cooperative Coevolution (CC) [19]. Like the GA or GP algorithm, the CC algorithm explores the solution space in a random fashion. As in GA, the CC algorithm applies the operators crossover, mutation, and selection to generate potential solutions. However, in CC, the representation of a solution is broken down into sub-parts, each of which encodes part of the solution and is evolved separately. Therefore, sub-populations are generated and maintained in each generation of the CC algorithm. In this manner, the opportunities for searching and exploring different solution subspaces are increased.

By comparing the algorithms in Figure 3, it is clear that the major difference between these two models lies in how the evaluation of individuals is performed. As stated earlier the evaluation in the GA model is performed on an individual (as a whole) in a population; on the other hand, in the CC model, individuals from separate sub-populations must come together to create an "organism" that is viewed as the solution. Hence, in the CC model, an individual cannot provide a meaningful solution to the problem and requires the cooperation of individuals from other sub-populations.

[1] Other representations such as graphs and Lists are also possible for GP's.

```
Procedure GA_GP( )
initialize population
loop
  evaluate individuals
  store best individual

   select mating candidates
   recombine parents and use their
  offspring as the next generation
until stopping condition

return best individual
```

```
Procedure CC_GP( )
initialize subpopulations
loop
evaluate organisms (solutions)
store best organism
for each subpopulation
  select mating candidates
  recombine parents and use their
   offspring as the next generation
end for
until stopping condition
return best organism
```

Figure 3. Comparing a typical GA and CC algorithm.

The success of CC depends on 4 criteria: (a) Problem decomposition, (b) Interdependability, (c) Credit assignment, and (d) Population diversity. These criteria are described in the following section when the concept of super agent will be discussed.

4.2 Agents and Super-Agents

In order to evolve useful programs, programs that can detect anomalies and act as an IDS, we must define how an agent is constructed first. As mentioned earlier, an agent is a code fragment, possibly a program. A suitable evolutionary model for evolving programs is GP. GP alone can be used to create IDS codes or agents. An advantage of applying GP to create IDS agents is that in this paradigm the agents can be adaptable to new, unseen intrusions, one only needs to train new agents, once a new intrusion occurs. Because so many agents can exist, possibly in parallel, this provides a distributed environment where fault tolerance is inherent. In addition, the audit trail generates a great amount of data that can be utilized for both training and testing the agents. Although these advantages make the GP-only based IDS a very attractive and viable approach, it does have two major shortcomings, 1) there are no interactions between agents, 2) this approach fails to treat security as a system wide problem.

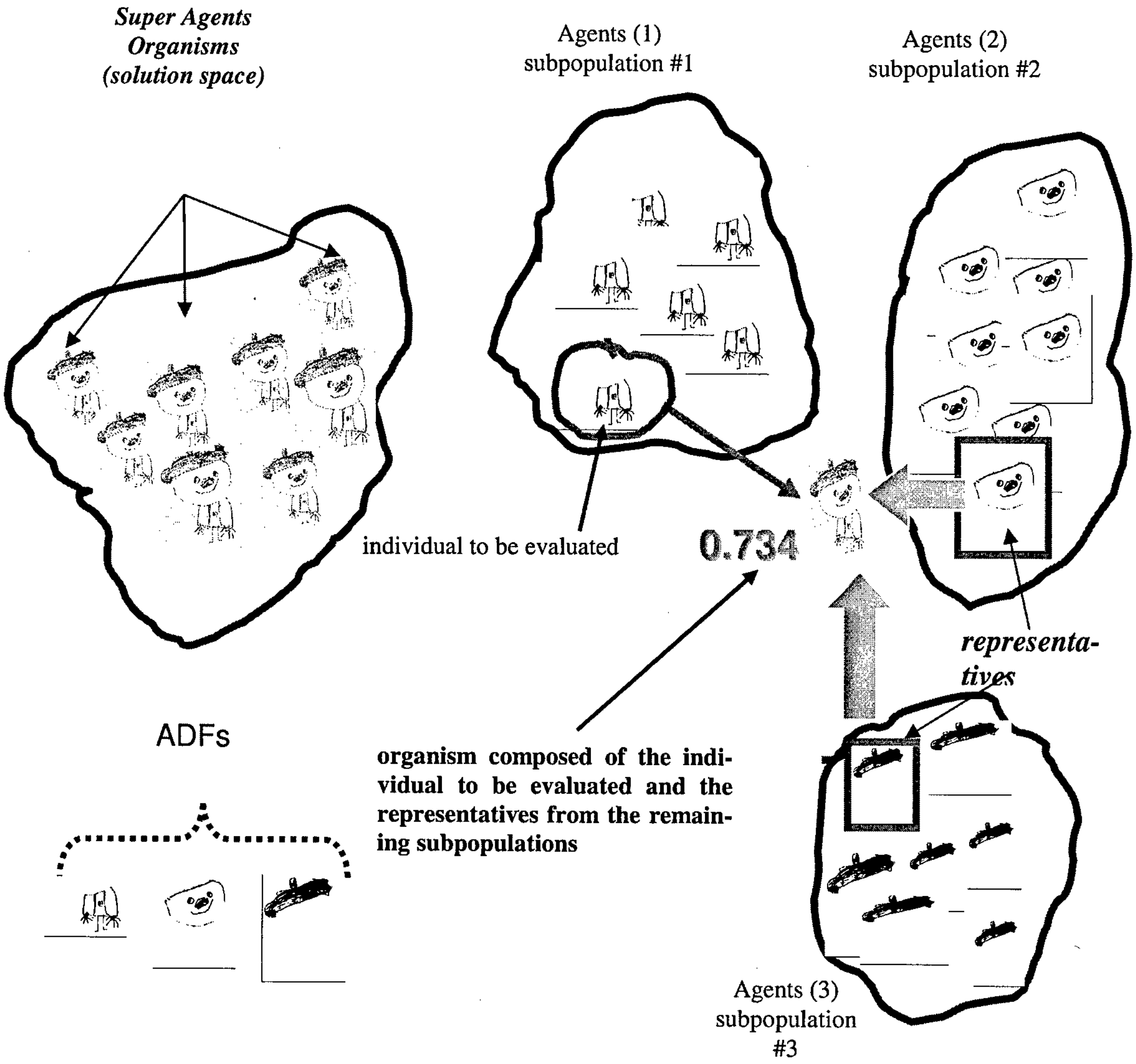
As indicated in Figure 1 agents are combined through a multiplexer; this interaction at the level of the multiplexer is a predefined design, which requires human input and bias and once the multiplexing criterion is selected, it remains static throughout the lifetime of the IDS system. Moreover, because an agent is evolved separately, and as a whole, different runs of the GP do not guarantee achieving separate agents that may detect anomalous behavior significantly better than each other.

To overcome these problems, a new type of agent, *Super Agent*, is introduced, see Figure 4 and Figure 5. Just like the agent described earlier, a super agent is an agent that is made of many mini-agents. A mini-agent or simply an agent, is a simple function or routine that evolves. An ADF is an automatically defined function, it is essentially a subroutine that is evolved separately from a parent tree, this enables the generation of type-safe parse. A simple function itself may or may not be a solution but many functions together construct a full program that acts as the IDS system. Therefore a super agent is a fully functional program made of several functions or subroutines. By providing the concept of the super-agent and decomposing the problem into mini agents, we have met the first criteria for applying the cooperative coevolutionary method for constructing an IDS system. The other criteria's are also met as follows:

(a) our representation, single Automatic Defined Functions ADFs, provides interdependencies between subcomponents,

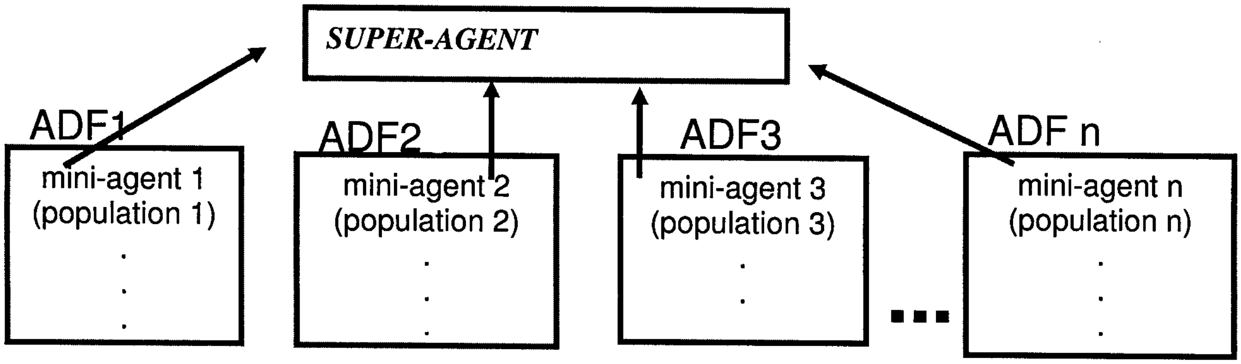
(b) the objective or fitness function to evaluate each ADF is the suspicion value each super agent evaluates to.

(c) population diversity can be maintained by roulette wheel selection method.



Super Agents Organisms (solution space)
Agents (1) subpopulation #1
Agents (2) subpopulation #2
individual to be evaluated
0.734
representa-tives
ADFs
organism composed of the individual to be evaluated and the representatives from the remaining subpopulations
Agents (3) subpopulation #3

Figure 4. A graphical example of a super agent formed by evolving individual agents. Note that each agent is defined as an Automatic Defined Function or ADF.



SUPER-AGENT
ADF1
ADF2
ADF3
ADF n
mini-agent 1 (population 1)
mini-agent 2 (population 2)
mini-agent 3 (population 3)
mini-agent n (population n)

Figure 5. A super agent (an IDS system) is made of many mini-agents (e.g., standalone functions).

An important part of the evolutionary algorithm is the evaluation function, referred to as the fitness function. A suitable fitness function must be provided that enables the search process to continue in the direction of a suboptimal solution. The fitness function for the agent code is given as follows. First we define a suspicion value as the value reported by the super-agent as how suspicious the activity under investigation is. For a given scenario, the absolute difference between the super agent's reported suspicion and the scenario's outcome is the fitness function as follows:

$$\delta = |outcome - suspicion|$$

where δ is the absolute difference between the agent's reported suspicion and the scenario's outcome. The smaller the δ, the better the agent has predicted the outcome. For a set of n training scenarios, the fitness is:

$$fitness = \frac{\sum_{i=1}^{n} \delta_i}{n}$$

The agent with the lowest fitness value is selected for next generation. Once the evolution is stopped (e.g., after a number of iterations, or achieving certain fitness value), the best mini-agents (functions) are selected and combined to create the super-agent.

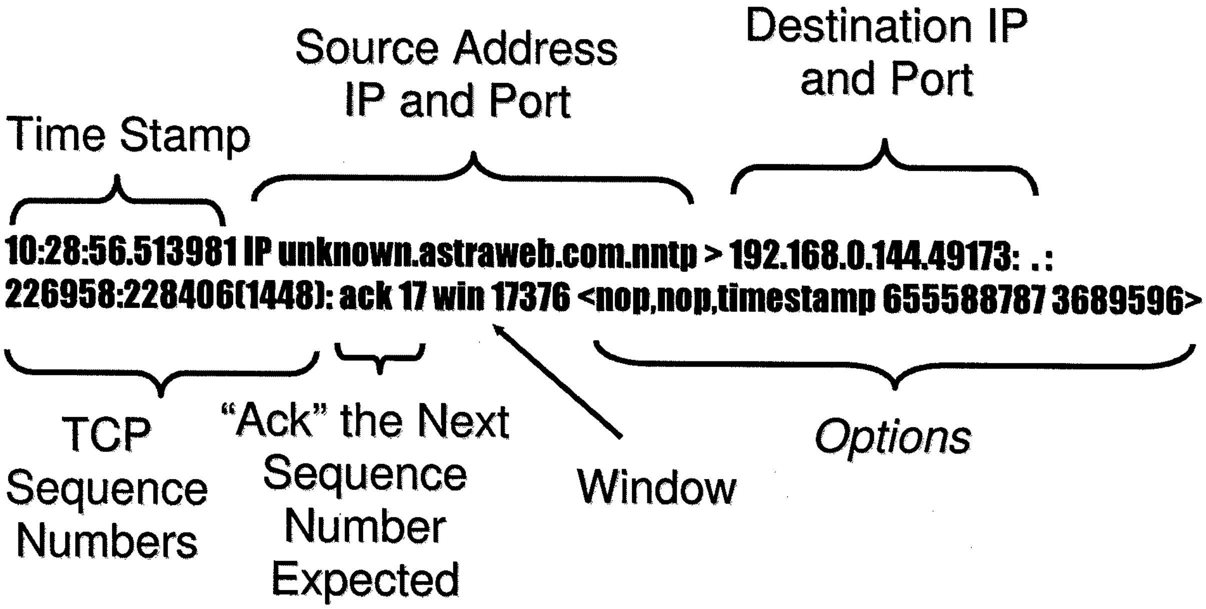
3.4.3 Metrics – Features: Two types of intrusions may occur in a computer system, one that is initiated from outside the computer and requires a network or other form of connection and the other initiated internally where, for example, a user consciously or unconsciously may affect the confidentiality, integrity or availability of information and resources of the computer. In either case, we assume that audit trails are in place where a snap shot of the computer system is stored. An audit trail is a log file generated by a specific program such as tcpdump or weblog, which monitor TCP (Transmission Control Protocol) connection or web-accesses respectively. Variety of programs for generating audit trails exists [20, 21]. Since majority of intrusions are initiated externally, we focus on the network intrusions where a user must first gain access to the resources or the computer on a remote location through a network connection. As mentioned earlier, tcpdump is the tool of choice. Tcpdump operates in the following layers:

Application: tcpdump analyzes application level protocols such as NFS, Telnet, FTP , Samba and others.

Transport: tcpdump intercepts and analyzes TCP and UDP packets.

Internet: tcpdump intercepts and analyzes ICMP packets.

Network Access: tcpdump can intercept and analyze many network level protocols such as ARP which is an Ethernet protocol.



```
10:28:56.513981 IP unknown.astraweb.com.nntp > 192.168.0.144.49173: . :
226958:228406(1448): ack 17 win 17376 <nop,nop,timestamp 655588787 3689596>
```

Figure 6. Example of a TCPDUMP outputline.

Examples of misue by layers are given as follows:

Application: Monitoring the Application Layer can expose suspicious or unwanted use of Network Services, Instant Messengers, P2P programs, or vulnerabilities in applications.

Transport: Manipulation of TCP/UDP packets can be used to DOS networks (Denial of Service).

Internet: ICMP Packets can be employed to DOS a machine by requesting connections rapidly, overwhelming the resources of the target machine.

Network Access: Switches can be flooded with fake MAC address announcements overwhelming their resources, and forcing them to revert to hub mode – making their traffic easy to intercept.

Tcpdump outputs lines of human readable text, which represent the TCP (Transport Control Protocol) header, see Figure 6.

Utilizing the tcpdump format, we obtain a wealth of features. The features are described in details in Appendix A.

3.4.4 System Architecture: Figure 7 describes the system architecture for the super-agent. The system uses the audit trail data from the TCPDUMP data (see Figure 6). The audit trail can be tailored to specific system component, for example if web-access is of interest, the audit data can come from a web-log, or if user behavior is being tracked, a user profile can supply the audit trail. The data is in raw format and it must be parsed and appropriate features extracted from it. Once the features are derived, the abstraction layer must parse the features for the appropriate types. Each ADF is a specific subroutine, a function that operates on a specific data type. The number of ADF's (or agents) to use is of particular interest. One can apply n features to the system, requiring n agents to be evolved; however, it may be more efficient to apply a subset of features so that Super agents can be specialized for a certain task. For example, if port-scanning intrusion is to be detected, then the count features can be utilized (see next section for the features). The CC-GP module initializes the agents and through the evolutionary process described in Figure 3, and Figure 4, evolves the agents. The final result is a linear combination of these agents which form a program containing several functions (more specific, function calls). The super agent is then input to a production system on the computer. As the audit trails are gathered and the super agent is performing its task, new audit trails generated can be used to update or create new super agents. The feedback from the computer system enables the super agents to deal with new threats. If a threat has been missed, the agents can evolve again based on the new audit trail information (training data). If a new threat has been assessed and the agents have been unable to detect it, a new super agent can be evolved.

The *window size* in the Feature derivation also plays an important role. The window size assures the granularity of the agents is preserved. For example, a port scan attack can be a burst attack in which a port scanner attempts to quickly scan the system ports or it can be more methodical and distributed where a port scan can occur over a long period of time. In this manner, unless an IDS has long term planning, it will not be able to detect such attacks. In our architecture we can track several window sizes of seconds to hours and days; therefore, potentially updating different agents.

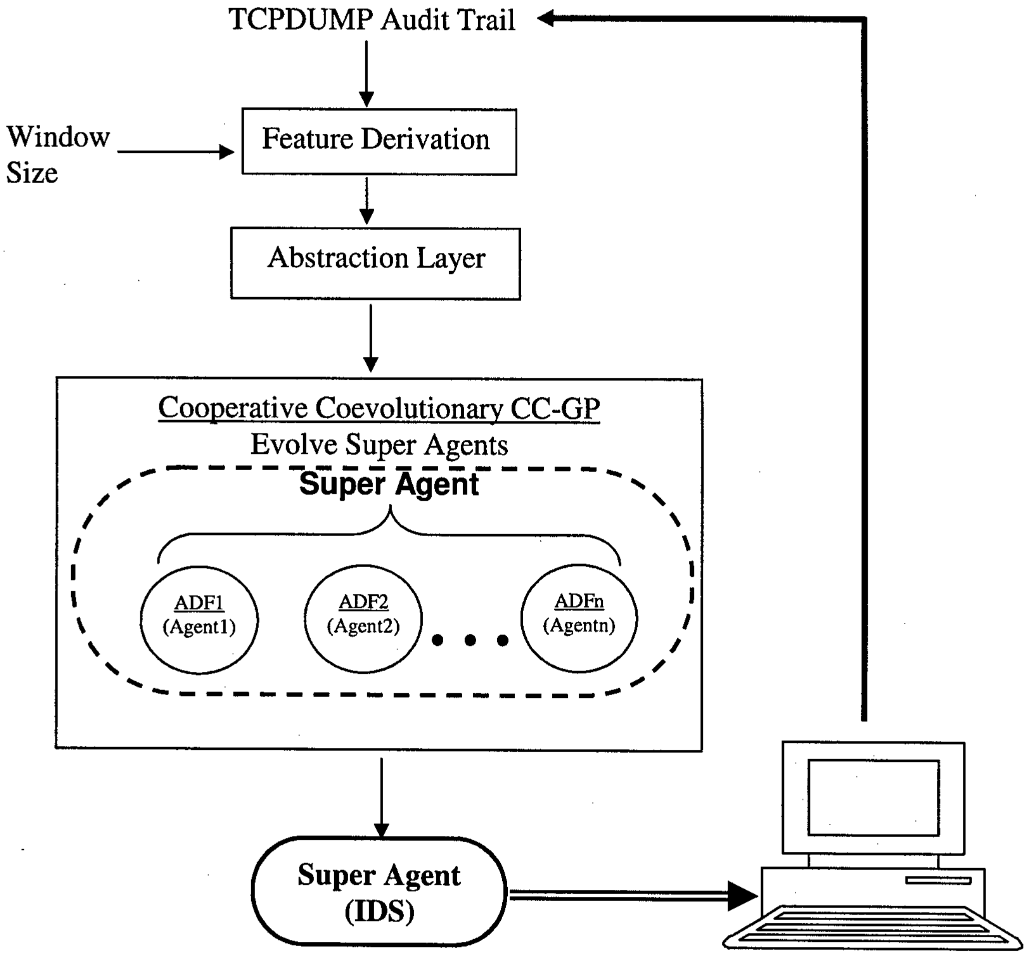


Figure 7. System Architecture: Input from the Audit Trail disseminates to the CC-GP module for evolving agents and super agents.

3.5 Features Derived from TCPDUMP Audit Trail

Terminology

Window	A unit of time in which events are grouped together for analysis.
Count	Number of times an event occurs per unit time (Window)
Density	The percentage of a whole a certain set of events represents.
Rate	Rate of change of a feature

<u>Feature Types</u>

Port	is a number assigned to a process or service type. Type: Integer
Time	is the time given in Hour:Min:Sec.Fraction of a second
IP_{source}	The IP address of the source
IP_{dest}	The IP address of the destination

Packet Type: identifies the type of protocol used for the packet, values include: UDP, TCP, ARP, ICMP. Other values are possible and are enumerated in the ip(4) man page, however the aforementioned packet types are adequate to begin. In the future we will consider protocol analyzers as well.

Flags: indicates situation for connection, only valid for TCP packets:

S	Syn Packet, host wishes to establish connection.
ACK	Host acknowledges having received data.
f (Fin)	Sender intends to close connection
r (Reset)	Sender aborts connection immediately
p (Push)	Sender sends data immediately.
urg (Urgent)	Packet has high priority.

Derived Features: The following are features based on the individual fields in TCPDump. Each feature is precisely defined. The values from all these features can be derived from the TCPDump file. Some features are time based and require a window (time frame). The resolution of this window will be decided later but some suggestions are window of a minute, hour, day, week and month size. Having different window sizes can help in detecting distributed attacks. Basically a window can be 1 minute, 10 minutes, 2 hours, 1 day, 1 week, etc. In this manner each window can be considered as a feature, for example, short-term features vs. long-term features.

"Count" Features: Count the occurrence of a specific event.

Type: Unsigned integer

Example:

Time (window=1)	IP Address	Count
09-10-2004 12:01:01	66.33.12.14	1118
09-10-2004 12:01:02	66.33.12.14	2312
09-10-2004 12:01:02	38.72.142.1	1942
...	...	...

IP Count

Definition: Given an IP address (of the form xxx.xxx.xxx.xxx, where 0 ≤ xxx ≤ 255 is a one byte integer), the IP Count, measures how many times an IP address has occurred per WINDOW. (NOTE: Source and Destination can be treated as different features).

Port Count

Definition: How many times a particular port (e.g., ssh=22) was accessed per WINDOW. There are currently 65536 ports to be scanned. A list of important or active ports can be maintained so calculating the value for this feature can be done more efficiently.

Packet Type Count

Definition: This is the number of times a particular packet type (see packet types in TCPDump) shows up within a WINDOW time frame.

Event Count

Definition: How many TCPDUMP events occur per window.

“Unique Count” Features

Unique Count features are counts of unique events.

Type: unsigned integer

Example:

Time	**Count**
09-10-2004 12:01:01	**25**
09-10-2004 12:01:02	**27**
09-10-2004 12:01:03	**22**
…	**…**

Unique Ports Count

Definition: Describes how many different ports are accessed per WINDOW. (For example, if within 1 minute WINDOW, ports 11, 22, and 25 are the only ports accessed the value for this feature will be 3).

Unique IP Count

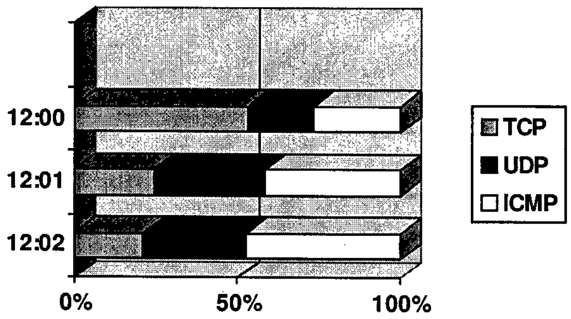
Definition: Describes how many different IP's are accessed per WINDOW. (For example, if IP: 127.27.63.1 and 64.68.127.1 are the only IP's appeared within a window, then the value of this feature will be 2)

"Density" Features

Density features track the percentage of a WHOLE for each component of a count feature.

Type: Multiple floats

Example:



Packet Density

Definition: Percentages of all packets received per window of a certain type.

IP Density

Definition: Percentages of all packets received by a certain IP.

Port Density

Definition: Percentages of all packets addressed to or from a certain IP.

Flags Density

Definition: Percentages of all packets received containing flags.

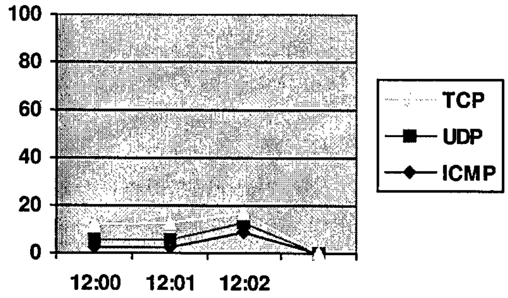
"Density Rate" Features

A quantification of the rate of change of a particular density feature. It is alculated by:

$$\frac{X2 - X1}{\text{Window}}$$

Type: Multiple Float Representing Percentages

Example:



Packet Density Rate

Definition: Rate of change of each component of the Packet Density.

IP Density Rate

Definition: Rate of change of each component of the IP Density.

Port Density Rate

Definition: Rate of change of each component of the Port Density.

Flags Density Rate

Definition: Rate of change of each component of the Flags Density.

Methods for Calculating Window Count Rate

Terminology:

Dictionary - *Allows one to associate an object with another. Also called: Map, and Associative Array*
Key - *An object used to retrieve another in*
List - *An extensible array*
Bucket - *A data structure (on our case a list) which similar items are grouped into.*

Proposed data structure: A dictionary of lists. Events are stored in "buckets" (lists) which are associated in a dictionary to a key which takes the form of a string literal "DATE, TIME" where DATE or TIME are rounded to the WINDOW size. For instance: in a 5 minute window the Keys would look like:

"10-10-2004, 12:00:00"
"10-10-2004, 12:05:00"
"10-10-2004, 12:10:00"
A 24 hour window:
"10-10-2004, 00:00:00"
"10-11-2004, 00:00:00"
"10-12-2004, 00:00:00"

Keys can be associated with either lists of events that fall into that bucket (very expensive RAM use) from which statistics can be generated. Or more likely, statistics will be generated from data as it is received, and data structures which represent this aggregated data will be stored (pro: efficient RAM use, con: can't query individual events).

3.6 Conclusions

Intrusion detection remains a challenging problem in the computer security domain. Several different approaches for misuse and anomaly detection have been proposed; however, as attacks become more sophisticated and distributed, more lightweight and distributed detection systems are necessary for tracking intruders. An evolutionary based agent oriented IDS system was introduced. The architecture described is capable of adapting to new treats. The features derived also enable the system to track both short term and long-term intrusions. This capability is extremely important in today's computer security problem since more sophisticated attacks occur over longer periods of time; therefore, distinguishing them from the normal behavior of the system becomes a very challenging problem.

The system above is currently being constructed as a research testbed. For future research in this area, topics of interest include feature selection, frequency of agent update, the interaction of super agents in a large scale system, and parallelization of the architecture.

References:

[1] D. Longley and M. Shain, **Data and Computer Security: Dictionary of Standards, Concepts and Terms**, Stockton Press, 1987.

[2] D. Russell and G. Sr. Gangeni, **Computer Security Basics**, O'Rielly and Associates Inc., 1991.

[3] S.E. Smaha, "Haystack: An intrusion detection system," *4th Aerospace Computer Security Applications Conference*, pp. 37-44, Dec. 1988.

[4] T. Lunt, "A survey of intrusion detection techniques," *Computers and Security*, Vol. 12, No. 4, pp. 405-418, June 1993.

[5] K. Ilgun, USTAT – A Real-Time Intrusion Detection System for UNIX., Master's Thesis, University of California at Santa Barbara, Nov. 1992.

[6] N. Ye, "A Markov chain model of temporal behavior for anomaly detection", *Proceedings of the 2000 IEEE Systems, Man, and Cybernetics Information, Assurance and Security Workshop*, 2000.

[7] K. Yamanishi, J. Takeuchi, G. Williams, , and P. Milne, "On-line unsupervised oultlier detection using finite mixtures with discounting learning algorithms," *In KDD*, pages 320–324, 2000.

[8] H. S. Teng, K. Chen, and S.C. Lu, " Security audit trail analysis using inductively generated predictive rules," *Proc. of the 11th National Conference on Artificial Intelligence Applications*, pp. 24-29, Mar. 1990.

[9] S. Kumar and E. Spafford, An Application of Pattern Matching Intrusion Dectection. Technical Report 94-013, Purdue University, Department of Computer Sciences, Mar. 1994.

[10] J. Ryan, M.J. Lin, and R. Miikkulainen, "Intrusion detection with neural networks," *Proceedings of AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management*, pages 72–77, 1997.

[11] E. M. Knorr and R. T. Ng. "Algorithms for mining distance-based outliers in large datasets", *Proc. 24th Int. Conf. Very Large Data Bases, VLDB*, pages 392–403, 1998.

[12] M. Joshi and V. Kumar, "Credos: Classification using ripple down structure (a case for rare classes)," *Proc. of 19th International Conference on Data Engineering*, 2003.

[13] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *Proc. of the ACM SIGMOD Conference*, pp. 427–438, 2000.

[14] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," *IGMOD Conference*, 2001.

[15] M. Breunig, H. P. Kriegel, R. T. Ng, and J Sander, "Lof: Identifying density based local outliers," *Proc. of the ACM SIGMOD Conference*, 2000.

[16] P. Helman, J. Bhangoo, "A statistically based system for prioritizing information exploration under uncertainty," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 27, No. 4, pp. 449-466, 1997.

[17] M. Crosbie and E. Spafford, Defending a Computer System Using Autonomous Agents, Technical Rport, CSD-TR-95-022, Department of Computer Sciences, Purdue University, 1995.

[18] M.A. Potter and K.A. DeJong, "A cooperative coevolutionary approach to function optimization," *Proc. of the 3rd Conference on Parallel Problem Solving from Nature*, pp. 249-257, 1994.

[19] Rebecca Gurley Bace, **Intrusion Detection**, McMillan Technical Publishing, 2000.

[20] Stuart McClure, Joel Scambray, and George Kurtz, **Hacking Exposed: Network Security Secrets and Solutions**, 4th Ed., McGraw Hill, 2003.

APPENDIX A

APPENDIX A

This appendix summarizes useful links, research and commercial IDS systems.

Links of interest

IDS Buyers Guide (sponsored by industry):
http://www.ipa.go.jp/security/fy11/report/contents/intrusion/ids-meeting/idsbg.pdf
Cert Intrusion Detection Checklist:
http://www.cert.org/tech_tips/intruder_detection_checklist.html
Center for Research in Information Security
http://www.cerias.purdue.edu/research/
http://www.cerias.purdue.edu/about/history/coast_resources/intrusion_detection/
Introduction to Distributed IDS:
http://www.securityfocus.com/infocus/153
Attack Signature Database:
http://www.whitehats.com/ids/
Paper: Data Mining Approaches for Intrusion Detection
http://www1.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html
List of IDS papers:
http://www.cc.gatech.edu/~wenke/ids-readings.html
Paper comparing GP to other IDS methods:
http://www-2.cs.cmu.edu/~cheeko/intrusion/
An intelligent host based IDS
http://people.roqe.org/kr/docs/ml-ids-talk.pdf
Intrusion Detection

In this project we investigate learning (data mining) techniques for building models that can detect intrusions. To detect unseen attacks, we currently focus on anomaly detection. Our models are built based on data gathered from the network and operating systems. We have audit data provided by DARPA that contain normal and attack activities. Long-term goals include cost-sensitive modeling and correlation among distibuted models.
http://cs.fit.edu/~pkc/id/

Key

[C]	Commercial Product
[O]	Open Source Product
[R]	Research Product
[F]	Features
[D]	DataSheet
[H]	Homepage
[M]	Comments
[P]	Synopsis Provided by website
[HAR]	Hardware Based

[SOF] Software Based
[NET] Network Sniffing Based IDS
[LOG] Log Based IDS
[STA] Statsistical Based
[SIG] Signature Based
[DIS] Distributed IDS

[C] NFR Intellgient IDS [NET][LOG][SIG]
[H] http://www.nfr.com/solutions/system.php
[P] With NFR, you get a uniquely clear and precise view of suspicious activity on your network. A combination of pattern matching and anomaly detection identifies both known and unknown types of attacks. Customization and fine-tuning enables you to detect "real attacks" and minimize distracting false positives. Comprehensive reporting and analysis tools help you to prepare for a future attack, or prepare for criminal proceedings.
[M] Claims to be a learning IDS but features indicates signature only.

[C] Still Secure BorderGuard [NET][HAR][SIG]
[H] http://www.stillsecure.com/products/bg/index.php
[D] http://www.stillsecure.com/docs/StillSecure_BG_datasheet.pdf
[M] Seems unremarkable, no mention of any technology used, just vague references to learning abilities.

[C] Mcafee Intrushield [NET][SIG]
[H] http://www.anidirect.com/products/intrusionprevention/networkintrusion.html
[D] http://www.anidirect.com/products/intrusionprevention/ds_intrushieldidssensor.pdf
[M] Hardware based, signature based IDS.

[O] Prelude [NET][LOG][SIG]
[H] http://www.prelude-ids.org/index.php3
[F] http://www.prelude-ids.org/rubrique.php3?id_rubrique=24
[M] GPL'd signature based software IDS, extensible, capable of monitoring wide range of hardware/software products such as WWW, SQL, FTP servers and router logs.

[R] Columbia IDS [NET][DIS]
[H] http://www1.cs.columbia.edu/ids/
[P] This project is a data-mining based approach to detecting intruders in computer systems. The project approaches the intrusion detection problem from a data-mining perspective. Large quantities of data are collected from the system and analyzed to build models of normal behavior and intrusion behavior. These models are evaluated on data collected in real time to detect intruders.
[M] Seems to monitor network and machine activity, including filesystem data and registry activity (trivial), however web pages haven't been updated in two years and no source code or project status is available. Has nice publications list at: http://www1.cs.columbia.edu/ids/library/

[O] Firestorm IDS
[H] http://www.scaramanga.co.uk/firestorm/index.html

[M] Open source NIDS, only sensors so far, no analysis

[O] SID [HOS]
[H] http://sid.sourceforge.net/
[P] Shell/PTY Intrusion Detection: Aims at detecting unwanted PTY action on UNIX systems. SID-IDS is a Host Intrusion Detection System. Consists of a kernel part and a user part. The kernel part plugs into terminal processing subsystem and logs hashed terminal lines. The user part reads log entries (hashes) and takes appropriate action upon finding unexpected log entries.
[M] Contains examples of shell monitoring!

[O] Snort [NET][SIG]
[H] http://www.snort.org/
[M] Signature Database: http://www.snort.org/snort-db/
[M] IDS Resources: http://www.snort.org/docs/

[O] LIDS [HOS]
[H] http://www.lids.org/
[P] The **Linux Intrusion Defence System** (LIDS) is a kernel patch and admin tools which enhances the kernel's security by implementing Mandatory Access Control (MAC). When it is in effect, chosen file access, all system network administration operations, any capability use, raw device, memory, and I/O access can be made impossible even for root. You can define which programs can access specific files. It uses and extends the system capabilities bounding set to control the whole system and adds some network and filesystem security features to the kernel to enhance the security. You can finely tune the security protections online, hide sensitive processes, receive security alerts through the network, and more. LIDS currently support **kernel 2.6, 2.4**. LIDS is released under **GPL**.
[M] Very interesting idea, doesn't seem to be an IDS except that it can report abuse

[C] Cisco IDS [SIG][NET]
[H] http://www.cisco.com/warp/public/cc/pd/sqsw/sqidsz/
[M] Unremakrable network IDS

[C] Next-Generation Intrusion Detection Expert System (NIDES) [SIG][HOS]
[H] http://www.sdl.sri.com/projects/nides/whatisnides.html
[P] NIDES performs two types of analysis. Its statistical analysis maintains historical statistical profiles for each user and raises an alarm when observed activity departs from established patterns of use for an individual. The historical profiles are updated regularly, and older data "aged" out with each profile update, so that NIDES adaptively learns what to expect from each user. This type of analysis is intended to detect intruders masquerading as legitimate users. Statistical analysis may also detect intruders who exploit previously unknown vulnerabilities who could not be detected by any other means. Statistical anomaly detection can also turn up interesting and unusual events that could lead to security-relevant discoveries upon investigation by a security officer. The statistical analysis is customizable: several parameters and thresholds can be changed from their default values, and specific intrusion-detection "measures" (the aspects of behavior for which statistics are kept) can be turned on or off.

[R] MAIDS: Mobile Agent Intrusion Detection System
[P] Feature Selection Using a Genetic Algorithm for Intrusion Detection (postscript) (full citation)
[H] http://latte.cs.iastate.edu/Research/Intrusion/
[M] Research IDS

[R] AFIDS Autonomous Agents for Intrusion Detection
[H] http://www.cerias.purdue.edu/about/history/coast/projects/aafid.php
[M] A library for sensor construction

[R] Machine Learning for Intrusion Detection
[H] http://ida.first.fhg.de/projects/mind/mind_official/
[M] research project with no Online Documentation Whatsoever

[R] MINDS - Minnesota Intrusion Detection System
[H] http://www.cs.umn.edu/research/minds/
[P] The overall objective of this research is to develop high performance data mining algorithms and tools that will provide support required to analyze the massive data sets generated by various processes that monitor computing and information systems. This research is being conducted as a part of MINDS (Minnesota Intrusion Detection System) project that is developing a suite of data mining techniques to automatically detect attacks against computer networks and systems.

[R] *A Data Mining Approach for Building Cost-sensitive and Light Intrusion Detection Models*
[H] http://www.cc.gatech.edu/~wenke/project/id.html
[M] Attackers can use automated scripts to generate a very high volume of intrusions to overwhelm an IDS and its operational staff, and then launch the intended and more serious attacks which may now go undetected. The limited time and resources therefore need to be focused on detecting the most damaging intrusions. In other words, a high statistical accuracy should not be the main goal of an IDS; rather, the more important goal should be the maximum reduction in intrusion damage cost with minimum IDS operational cost. The objective of this project is to study the theoretical foundations and the development approaches for cost-sensitive intrusion detection systems. In particular, we are focusing on: study of the cost factors, cost models, and cost metrics related to intrusion detection; development of automated techniques for building cost-sensitive models that are optimized for user-defined cost metrics; and design of a system architecture for dynamically activating and configuring light intrusion detection modules that each specializes for a set of similar intrusions.

[R] ADAM - Audit Data Analysis and Mining
[H] http://www.ise.gmu.edu/~dbarbara/adam.html
[M] ADAM aims to implement an intrusion-detection software that uses a multistrategy approach along the following lines: 1. Detect events and patterns directly expressed by the operator of the system: the operator, being the ultimate entity responsible for the detection of the system is allowed to specify situations that she considers ``abnormal." The system monitors the audit trail for these conditions and alarms the operator. 2. Mine for association rules that are becoming fre-

quent recently and are not usually that frequent in similar circumstances (day of the week, time of the day). In order to do this, two things must be done: a. Mine the audit trail for the association rules that are becoming ``hot" in recent times (the window of observation being a tunable parameter), and b. compare those association rules with those that have been frequent at similar times in the past. Thus, a repository of ``aggregated" past rules is needed. 3. Use other means of data mining to uncover suspicious or abnormal patterns of behavior. 4. Filter and prioritize alarms to avoid flooding the operator during and actual intrusion. This step also has the purpose of minimizing the number of false diagnoses.

[R] STAT
[H] http://www.cs.ucsb.edu/~rsg/STAT/projects.html
[M] The STAT project provides a framework that supports the development and the control of scenario-based sensors. By using STAT, it is possible to create a set of sensors that will operate in different domains and environments, e.g., network-based sensors, host-based sensors, and application-based sensors.

[R] Darpa Intrusion Detection Evaluation
[H] http://www.ll.mit.edu/IST/ideval/
[M] Provides tools, tests, and data sets to evaluate IDS systems

[R] AID: Application Intrusion Detection
[H] http://www.cs.virginia.edu/~jones/IDS-research/
[M] Concept Intrusion detection has traditionally been performed at the operating system (OS) level, but OS intrusion detection systems (IDS) are frequently insufficient to catch internal intruders. We hypothesized that application specific IDS (AppIDS) could use the semantics of the application to detect more subtle, stealth-like attacks such as those carried out by internal intruders. We developed two extensive case studies to explore what opportunities exist for detecting intrusions at the application level, how effectively an AppIDS can detect the intrusions, and the possibility of cooperation between an AppIDS and an OS IDS to detect intrusions. The main conclusion was that an AppIDS can detect some intrusions that an OS IDS cannot thus increasing the overall effectiveness in detecting intrusions.

Chapter 4

Performance Modeling of a Vote-based Recognition System

Performance Modeling of Vote-based Object Recognition

Edwin S. Hong[a], Bir Bhanu[b], Grinnell Jones III[b] and Xiaobing Qian[b]
[a]University of Washington, IT/CSS Department, Tacoma, WA 98402
[b]Center for Research in Intelligent Systems, University of California, Riverside, CA 92521
e-mail: edhong@u.washington.edu; (bhanu, grinnell, xqian)@vislab.ucr.edu

ABSTRACT

The focus of this paper is on predicting the bounds on performance of a vote-based object recognition system, when the features are distorted by uncertainty in both feature locations and magnitudes, by occlusion and by clutter. A method is presented to calculate lower and upper bound predictions of the probability that objects with various levels of distorted features will be recognized correctly. The prediction method takes model similarity into account, so that when models of objects are more similar to each other, then the probability of correct recognition is lower. The effectiveness of the prediction method is validated in a synthetic aperture radar (SAR) automatic target recognition (ATR) application using MSTAR public data, which are obtained under different depression angles, object configurations and object articulations. Experiments show the performance improvement that can obtained by considering the feature magnitudes, compared to a previous performance prediction method that only considered the locations of features. In addition, the predicted performance is compared with actual performance of a vote-based SAR recognition system using the same SAR scatterer location and magnitude features.

Keywords: SAR data, performance prediction, automatic target recognition

1. INTRODUCTION

The goal of our research is to develop a formal framework for predicting the fundamental performance of any given object recognition task. This particular paper focuses on predicting performance bounds of model-based object recognition systems, where the recognition system uses a feature matching criteria that is vote-based. In the prediction problem, we are given a model database as well as criteria under which features in a model can be distorted. We predict the object recognition performance assuming the system is given test data features that are generated by taking a model from the database and subjecting it to the distortion criteria. Our distortion model includes uncertainty in both feature locations and magnitudes, occlusion and clutter.

Our prediction method is general enough to represent the behavior of a wide variety of vote-based recognition systems under many different kinds of distortion. It also takes model similarity into account, meaning that when models of objects are more similar to each other, then the predicted probability of recognition is lower. It can thus be used to explore how various distortion factors and model similarity affect object recognition performance. We believe that knowing how these factors affect object recognition performance is key to understanding and improving the effectiveness of object recognition systems.

Our prediction builds upon the previous work of Boshra and Bhanu.[1] We improve their previous performance prediction method by incorporating a more general feature set, namely by including magnitude as a component of the feature of a model. Although there have been other performance prediction methods, however none of them consider uncertainty, occlusion, clutter, and model similarity when generating the prediction.[1,2] In comparison

Further author information: (Send correspondence to Ed Hong)
Ed Hong: Address: Computing and Software Systems, University of Washington, 1900 Commerce Street, Box 358426, Tacoma, WA 98402, USA; Telephone: 1 253 692 4659

with Boshra and Bhanu's previous work, we also remove some of the assumptions that they needed to simplify the performance prediction computation.

In addition to building a general framework for performance prediction, we also validate our technique by testing it in a synthetic aperture radar (SAR) automatic target recognition (ATR) application using MSTAR public SAR data. These data are obtained under different depression angles, object configurations and object articulations. Experiments show the performance improvement that can obtained by considering the feature magnitude, compared to a previous performance prediction method that only considered the locations of features. In addition, the predicted performance is compared with actual performance of a vote-based SAR recognition system

Our preliminary results show that our prediction scheme may be an effective way to predict object recognition performance. We also show that the benefit of including the magnitudes of features enables us to get the same probability of correct recognition at a 5 to 10 percent higher distortion rate.

2. PREDICTION SYSTEM

In this section, we describe our system for predicting performance bounds for any vote-based object recognition system. Our prediction system takes as input a database of models each consisting of a set of features, and a distortion model. It then obtains performance bounds on the model database given the distortion model. Our features consist of a discrete x and y location, as well as a magnitude. Our distortion models uncertainty (in both location and magnitude), occlusion, and clutter. Our system can also handle arbitrary types of probability distribution functions (PDFs) for uncertainty and for clutter.

Note that the prediction results depends on the entire database of models given to it. When models in the database are more similar to each other, then our prediction for the probability of correct recognition (PCR) will be lower than if the models are less similar to each other. Our system generates a series of PCR bounds, one for each model M in the database. The PCR bounds for M represent the probability that M will be recognized correctly if it undergoes the given distortion, and the recognizer must choose from among the entire database of possible models for a result.

We first start by giving a mathematical description both vote-based object recognition and our distortion model. We then show how to compute the PCR bounds based on the mathematical descriptions.

2.1. Basic Object Recognition Definitions

Our general recognition system has a set $\mathcal{M} = \{M_1, M_2, \ldots, M_n\}$ of n models. Each model consists of a set of features, where each feature is a tuple of attributes. In our work, we have three attributes: a horizontal and a vertical attribute representing a discrete 2D location, and a magnitude. More formally, a feature f is a tuple (x, y, s), where $(x, y) \in \mathbb{Z} \times \mathbb{Z}$ is the location of the feature, and $s \in \mathbb{R}$ is the magnitude. We use $\mathcal{F}$ to denote the space of all features. Note that $M \in \mathcal{M}$ is an arbitrary set of features; each model M need not have the same number of features.

The recognition system also is given a set $\mathcal{T}$ of transformations over $\mathcal{F}$. Let $\tau \in \mathcal{T} : \mathcal{F} \to \mathcal{F}$ denote one such transformation. Then $\tau(f)$ represents the feature f' which results from taking feature f under transformation τ. In this work, we consider $\mathcal{T}$ = set of all integral 2D-translations. The magnitude s of each feature remains unchanged under our transformations. We use $\tau_{i,j}$ to denote the transform that maps (x, y, s) to $(x+i, y+j, s)$.

An object recognition system is given a set of features F, and seeks to identify $(M, \tau) \in \mathcal{M} \times \mathcal{T}$ that best describes F. We call the tuple (M, τ) a hypothesis. Intuitively, the input set F is generated by looking at the image data (or a simulated signature) of some model object, and then extracting a set of features from the data. Since the orientation and position of the object in the image data is unknown, the transform τ representing the location of the object needs to be found, in addition to identifying the correct model M. Furthermore, the feature data may be distorted, so that the match between F and some hypothesis (M, τ) will generally not be exact.

We consider vote-based recognition systems, where individual features that match contribute votes, and the hypothesis is picked based on the highest vote total. Consider the number of votes obtained in such a system, by some hypothesis (M, τ) on a set of features F. To allow for inexact matches, we consider any feature $f \in F$ to match a feature $f' \in M$ under transformation τ when $\tau(f)$ is "close enough" to f'. In general, there will be some region around each model feature f' that represents how close $\tau(f)$ needs to be in order to be considered a matching feature for f'; this region is known as *voting region* of f', denoted $\mathrm{VR}(f')$. The size of this region typically is chosen by the recognition system to improve performance.

One example definition of VR for a feature (x, y, s) is

$$\mathrm{VR}((x,y,s)) = \{(x',y',s') : (x',y') \in \mathrm{4NEIGHB}(x,y) \text{ and } .9s \leq s' \leq 1.1s\}, \tag{1}$$

where, $\mathrm{4NEIGHB}(x,y) = \{(x,y),(x+1,y),(x-1,y),(x,y+1),(x,y-1)\}$ is a set of neighbors of (x,y).

Strictly speaking, the voting region for a feature f_1 of M could be a completely different size and shape than that of feature f_2 of M. For this work, however, we will assume that the size and shape of each voting region of every feature of every model M in $\mathcal{M}$ is the same. In other words, there is just one function VR that, when applied to any feature f' of model M returns a region $\mathrm{VR}(f')$ centered around f'. Any transformed feature $\tau(f)$ falling in $\mathrm{VR}(f')$ is considered a match for f' under transformation τ.

Now let $V(F,(M,\tau))$ denote the number of votes generated by hypothesis (M, τ) on feature set F. We define this is as:

$$V(F,(M,\tau)) = |\{f' \in M : \exists f \in F, \tau(f) \in \mathrm{VR}(f')\}|$$

In other words, we are counting the number of features f' of model M that have at least one feature f in F matching it. Note that V, the number of votes, clearly depends on the exact definition for the voting region. To better distinguish between features, we might expect voting regions for two features f_1' and f_2' from some model M to be nonoverlapping. However, note that this is not a strict requirement of the above definition. If the voting regions of f_1' and f_2' overlap, then one feature f in the feature set F may potentially contribute two votes if it happens to be in the union of the two regions $\mathrm{VR}(f_1') \cup \mathrm{VR}(f_2')$.

We focus on the forced recognition problem, meaning that the object recognition algorithm $\mathcal{A}$ running on F returns the hypothesis (M, τ) that maximizes $V(F,(M,\tau))$.

The performance of the recognition algorithm is measured in terms of the percentage of correct responses when recognizing a test data set consisting of many feature sets. There is a ground truth associated with each feature set in the test data that represents the correct hypothesis. In many object recognition scenarios, a hypothesis that is close enough to the correct one is acceptable. Let $H_{acc}(F)$ be the set of acceptable hypotheses for the feature set F. Then the performance of algorithm $\mathcal{A}$ is measured by the number of feature sets F in the test data that satisfy $\mathcal{A}(F) \in H_{acc}(F)$.

2.2. Distortion Model

As part of the performance prediction scheme, we assume that the feature sets F input into the recognition algorithm are generated by distorting features of some model M in the model database. Our distortion process consists of three steps: uncertainty, occlusion, and clutter. The distortion depends on the following user-defined parameters:

- Uncertainty PDFs over $\mathcal{F}$ for each feature, representing how likely each feature is to be perturbed.

- Occlusion amount (O), for determining the number of features to occlude.
- Clutter amount (C), for determining the number of clutter features to add.
- Clutter Region (CR), for determining where clutter features should be added.
- Clutter PDF, for determining distribution of clutter over the clutter region.

For simplicity, we describe our distortion method using uniform PDFs. Using other PDFs is also possible and easy to implement.

Let $M = \{f_1, f_2, \ldots, f_k\}$ be the model to be distorted. Then the distortion algorithm $\mathcal{D}$ does the following:

1. (Uncertainty) Replace each $f_i = ((x, y), s)$, with a new feature f_i' chosen uniformly at random from the set
$$\{(x', y', s') : (x', y') \in \text{4NEIGHB}(x, y) \text{ and } .9s \leq s' \leq 1.1s\}$$
2. (Occlusion) Uniformly choose O features out of the k present (so each size k subset is equally likely); remove these features.
3. (Clutter) Add C additional features, where each feature is generated by picking a feature uniformly at random from CR (the clutter region).

The clutter region typically depends upon the given model M that we are distorting. As an example clutter region, consider the bounding box on the feature locations and magnitudes in M. More formally, let $x_{\max}$ ($x_{\min}$) represent the x value of the feature with maximum (minimum) magnitude in M. Similarly define $y_{\max}, y_{\min}, s_{\max}$, and $s_{\min}$. Then the bounding box clutter region CR is

$$\text{CR} = \{(x, y, s) : x_{\min} \leq x \leq x_{\max}, y_{\min} \leq y \leq y_{\max}, s_{\min} \leq s \leq s_{\max}\} \tag{2}$$

We define the *distortion region* of feature f, denoted by DR(f), as the union of all features that could be generated as uncertain versions of f. In the above algorithm, the distortion region is the set specified in step 1. Note that in Boshra and Bhanu,[1] the distortion region and the voting regions are always the same; they are both called the uncertainty region. Separating out these two regions is useful, because the recognition algorithm may not always know the size of the distortion region. Furthermore, under non-uniform distributions where the center of the distortion region is more likely to occur, having a voting region smaller than the distortion region may result in better performance.

In step 3 of this distortion model, it is possible that a clutter feature chosen is exactly the same as an occluded model feature. If this occurs, our clutter model adds less than C additional features that can be identified as clutter.

Furthermore, it is possible for two clutter features to be chosen with the same (x, y) location. In many cases, constraints on the method of generating test data prevents two features from ever sharing the same location, and possibly even from sharing in some cases may prevent features from sharing neighboring locations. In these cases, it is possible for our distortion algorithm to generate invalid distorted data, either due to overlapping distortion regions, or to clutter. We feel that adding a more sophisticated distortion model to handle these constraints will significantly increase the complexity of the prediction system without generating a noticeable improvement in results. We argue that the distortion regions are typically small enough that they rarely overlap, and when they do, the resulting effects are small. Furthermore, number of features studied is typically small compared to the size of the clutter region, making it unlikely that overlapping clutter will significantly affect our results.

2.3. Prediction Method Overview

In the performance prediction problem, our goal is to determine the performance of a given recognition system when the models to be recognized undergo distortion. Our goal is to determine the probability of correct recognition (PCR), which we define as follows:

Let PCR$(M, \mathcal{M}, \mathcal{A}, \mathcal{D}, H_{acc})$ denote the probability that recognition algorithm $\mathcal{A}$ will return a hypothesis in H_{acc} when run with knowledge of the models in $\mathcal{M}$ on an input generated by distortion algorithm $\mathcal{D}$ running on model $M \in \mathcal{M}$.

Note that $\mathcal{A}$ (described in 2.1) depends on the size of the voting region VR for each feature and the set of transforms $\mathcal{T}$. Also, $\mathcal{D}$ (described in section 2.2) depends on many distortion parameters, including occlusion and clutter. Note that the PCR implicitly takes into account the similarity between models by using its knowledge about the features of all models in $\mathcal{M}$.

Let $\hat{M}$ denote the probability distribution over feature sets obtained when applying $\mathcal{D}$ on model M.

We can now define $V(\hat{M}, (M', \tau))$ as a random variable representing the number of votes generated by $\mathcal{A}$ for distorted feature set $\hat{M}$ on hypothesis (M', τ).

The PCR can be formally defined as

$$\text{PCR}(M, \mathcal{M}, \mathcal{A}, \mathcal{D}, H_{acc}) = \Pr[\forall h' \notin H_{acc}, \exists h \in H_{acc}, V(\hat{M}, h) > V(\hat{M}, h')].$$

We write PCR(M, H_{acc}) to denote PCR$(M, \mathcal{M}, \mathcal{A}, \mathcal{D}, H_{acc})$ when $\mathcal{M}$, $\mathcal{A}$, and $\mathcal{D}$ are fixed, Similarly, we define the probability of incorrect recognition (PIR) as

$$\begin{aligned} \text{PIR}(M, H_{acc}) &= 1 - \text{PCR}(M, H_{acc}) \\ &= \Pr[\exists h' \notin H_{acc}, \forall h \in H_{acc}, V(\hat{M}, h) \leq V(\hat{M}, h')] \end{aligned}$$

Unfortunately, the abundance of hypotheses appears to make an exact computation of the PCR computationally infeasible. Thus we approximate the PCR by computing upper and lower bounds on it. For each hypothesis h and each model M, we first compute approximate PDFs for $V(\hat{M}, h)$. We then use these PDFs to derive upper and lower bounds on the PCR. Note that in our prediction algorithm, we compute $V(\hat{M}, h)$ only if it is necessary to compute our bounds; we omit the computation of PDFs for many infeasible hypothesis, such as those where $V(\hat{M}, h) = 0$ with probability 1.

2.4. Lower bound on PCR

When computing PCR(M, H_{acc}), we assume that the correct hypothesis $h_c = (M, \tau_{0,0})$ is in H_{acc}. We know that

$$\text{PIR}(M, H_{acc}) \leq \Pr[\exists h' \notin H_{acc}, \text{ s. t. } V(\hat{M}, h_c) \leq V(\hat{M}, h')],$$

because this probability above has less stringent conditions. This is a reasonable approximation since we expect h_c to get the most votes. We can now approximate the above by summing over probabilities of incorrect hypotheses having high numbers of votes:

$$\text{PIR}(M, H_{acc}) \leq \sum_{h' \notin H_{acc}} \Pr[V(\hat{M}, h_c) \leq V(\hat{M}, h')],$$

Now assuming that the voting region is bigger than or equal to the distortion region, we know $V(\hat{M}, h_c) \geq |M| - O$, where O is the amount of occlusion. This implies

$$\Pr[V(\hat{M}, h_c) \leq V(\hat{M}, h')] \leq \Pr[|M| - O \leq V(\hat{M}, h')].$$

Our overall lower bound on the PCR is

$$\mathrm{PCR}(M, H_{acc}) \geq 1 - \sum_{h' \notin H_{acc}} \Pr[|M| - O \leq V(\hat{M}, h')].$$

This is easily computed once PDFs for $V(\hat{M}, h')$ are known.

2.5. Upper bound on PCR

Let h_c represent the correct hypothesis $(M, \tau_{0,0})$. We first approximate the PCR by

$$\mathrm{PCR}(M, H_{acc}) \approx \Pr[\forall h \notin H_{acc}, V(\hat{M}, h_c) > V(\hat{M}, h)].$$

Note that this is exact and not an approximation when $|H_{acc}| = 1$. Now let H be a set of hypothesis, none of which are in H_{acc}. Then we can bound the above approximation by:

$$\Pr[\forall h \notin H_{acc}, V(\hat{M}, h_c) > V(\hat{M}, h)] \leq \Pr[\forall h \in H, V(\hat{M}, h_c) > V(\hat{M}, h)].$$

Now let us assume that for any two hypothesis h and h' in H, $\Pr[V(\hat{M}, h_c) > V(\hat{M}, h)]$ is independent from $\Pr[V(\hat{M}, h_c) > V(\hat{M}, h')]$. That means the above upper bound can be written as

$$\prod_{h \in H} \Pr[V(\hat{M}, h_c) > V(\hat{M}, h)]. \tag{3}$$

Similar to the method used in the lower bound, we can approximate the above upper bound as

$$\prod_{h \in H} \Pr[|M| - O > V(\hat{M}, h)]. \tag{4}$$

Following the work of Boshra and Bhanu,[2] we use the set of peak hypotheses, in the above bound. The peak hypotheses are those hypotheses whose expected number of votes (for $\hat{M}$) are local maxima in the space of transforms. It is formally defined as

$$\begin{aligned} H = \{(M', \tau_{i,j}) \quad : \quad & (M', \tau) \notin H_{acc} \text{ and } M' \in \boldsymbol{\mathcal{M}} \text{ and} \\ & \forall (x', y') \in \mathrm{8NEIGHB}(i, j), (x', y') \neq (i, j), \\ & \quad \mathrm{E}[V(\hat{M}, (M', \tau_{i,j}))] > \mathrm{E}[V(\hat{M}, (M', \tau_{x',y'}))]\} \end{aligned}$$

Here, $\mathrm{8NEIGHB}(i, j)$ is the set of 8 neighbors closest to (i, j) including (i, j), meaning $\{(x', y') : i - 1 \leq x' \leq i + 1 \text{ and } j - 1 \leq y' \leq j + 1\}$.

We argue that assuming the probabilities above are independent for peak hypotheses is a reasonable assumption, and not too far from the truth.

2.6. Computing $V(\hat{M}, h)$

Here we compute the random variable $V(\hat{M}, (M', \tau))$. For conciseness, we use V to denote $V(\hat{M}, (M', \tau))$; $\hat{M}$ and (M', τ) are dropped, since they are assumed to be fixed for the rest of this section. There are three components to the computation of V: First, we compute the PDF of V_s, a random variable representing the number of votes that (M', τ) receives due to model similarity, without considering occlusion. Then we compute the conditional PDF of V_o, representing the number of similar votes that are occluded given the distribution of V_s. Finally, we compute V_c, the number of votes contributed due to clutter. Let V_{so} denote the number of votes due to model similarity after considering occlusion. Then

$$V = V_{so} + V_c,$$

where V_{so} is given by

$$\Pr[V_{so} = x] = \sum_{v_s = x}^{|M'|} \Pr[V_s = v_s] \Pr[V_o = v_s - x | V_s = v_s].$$

We compute V_s as a sum of indicator variables X_i, where X_i is 0 or 1 depending on whether feature i of M, when perturbed according to the uncertainty PDF of the distortion model, will contribute a vote to hypothesis (M', τ). For the uniform distribution, $\Pr[X_i = 1]$ is very simple to calculate. Let f_i be feature i of M. Let $\mathrm{VR}(M')$ denote the union of all voting regions of all features of M'. Then $\Pr[X_i = 1] = \mathrm{VOL}(\mathrm{DR}(f_i) \cap \mathrm{VR}(M'))/\mathrm{VOL}(\mathrm{DR}(f_i))$, where VOL denotes the volume, calculated over both the location and magnitude.

For non-uniform distributions, the $\Pr[X_i = 1]$ is also fairly simple to calculate. It would simply mean calculating $\Pr[f_i \in \mathrm{DR}(f_i) \cap \mathrm{VR}(M')]$ according to the probability distribution given for the uncertainty in location and magnitude.

Note that we are summing over the distorted features in $\hat{M}$ to obtain our similarity count, and not summing over all possible voting regions that may contribute a vote. If there is no feature whose voting region overlaps with more than one distortion region, than our similarity count will be exact. If there are overlaps, then our method may over count by letting one voting region contribute more than one vote. However, we do not expect much overlap to occur in practice.

The conditional PDF of V_o is given by the hypergeometric distribution.

$$\Pr[V_o = x | V_s = v_s] = \mathrm{hg}(x, |M'| - v_s, O, |M'|),$$

where

$$\mathrm{hg}(x, n1, n2, N) = \binom{n1}{x} \binom{N - n1}{n2 - x} \Big/ \binom{N}{n2}.$$

Let p_C denote the probability that one clutter feature will contribute a vote. We calculate V_c as a sum of C random indicator variables $Y_1, \ldots, Y_C$, where Y_i is 1 with probability p_C. Given a uniform distribution for clutter (in both magnitude and location), we know

$$p_C = \mathrm{VOL}(\mathrm{CR} \cap \mathrm{VR}(M'))/\mathrm{VOL}(\mathrm{CR}),$$

where CR is the clutter region, and $\mathrm{VR}(M')$ is the union of the voting regions for each feature of model M'. Calculating p_C for non-uniform distributions is also relatively straightforward, by calculating a volume weighted by the PDF of the clutter region.

For example, assume we have a clutter PDF that is non-uniform over the location of features and uniform over magnitude. Let $w_{i,j}$ represent weight assigned for the clutter PDF at location (i, j). Since this is a PDF, we know $\sum_{(i,j)} w_{i,j} = 1$. Let $s_{\min}$ and $s_{\max}$ be the bounds for the clutter PDF magnitude at each location. Define $\mathrm{LOC} = \{(x, y) : \exists s, (x, y, s) \in \mathrm{CR} \cap \mathrm{VR}(M')\}$, as the set of possible locations for clutter. Then we would have the probability of clutter contributing a vote as

$$p_C = \frac{\sum_{(i,j) \in \mathrm{LOC}} w_{i,j} \, |\{s : s_{\min} \le s \le s_{\max}\} \cap \{(s : (i, j, s) \in \mathrm{VR}(M')\}|}{s_{\max} - s_{\min}}.$$

3. PRELIMINARY RESULTS

In order to validate the above object recognition performance prediction method, we apply it to identifying target vehicles from publicly available synthetic aperture radar (SAR) data. The data consists of images of vehicles at many different azimuths, taken from a 15 degree depression angle. Each image is first preprocessed to generate *scattering centers*, which are local eight-neighborhood peaks in magnitude of the SAR data images. In our prediction system, these scattering centers are the model features and each image of a vehicle at one particular azimuth corresponds to one model M. In the data sets we used, there are typically between 50 and 80 scattering centers generated in the image preprocessing step. We limit the number of features of each model to 30 by using only 30 scattering centers with the highest magnitude in each image. The scattering centers with higher magnitude correspond more to the detailed geometry of the object than those of lower magnitude.

The model database $\mathcal{M}$ that we use consists of 582 models: 194 images of a T-72 tank in configuration #132, 194 images of a BMP2 Infantry Fighting Vehicle in configuration #c21, and 194 images of a BTR-70 Armored Personnel Carrier in configuration #c71. Note that each image depicts a vehicle at a different azimuth.

In the preliminary results reported below, we used the following parameters in all cases:

- The voting region VR for each feature is the 4 neighborhood region around the feature, along with a 10% uncertainty in magnitude. In other words, it is the sample definition given in equation (1).
- There is only one acceptable correct hypothesis for each generated distorted feature set. In other words, when feature set F is a distorted version of model M, then the set of acceptable hypothesis is given by $H_{acc}(F) = \{(M, \tau_{0,0}\}$.
- Uncertainty PDFs in the distortion model are chosen uniformly at random over the uncertainty region, and the uncertainty region around each feature is the same as the voting region given in 1.
- The occlusion rate (O) is equal to the clutter rate (C). Thus, after occluding O features and adding C clutter features, the number of features for each distorted model (30) remains the same as for the original model. Hence forth, we use the term *Distortion Rate* to refer to both O and C.
- We use a slight modification of the bounding box clutter region given in equation (2). Our clutter region replaces the condition $s_{\min} \leq s \leq s_{\max}$ in equation (2) with $s_{\min} - 10 \leq s \leq s_{\max}$. Allowing features with lower magnitudes models the effect of having high-magnitude scattering centers obscured by clutter, and replaced by scattering centers that were not originally part of the top 30 scattering centers.
- The Clutter PDF in the distortion model is uniform.

Our first experiment calculates the upper and lower bound to the PCR for each model M, according to the method presented in sections 2.4, 2.5, and 2.6. To generate an average PCR figure for the entire database $\mathcal{M}$, we take the arithmetic mean of the PCRs for each model in the database. We compute the PCR bounds for each possible distortion rate value, from 10 up to the maximum of 80. This is shown in Figure 1.

In our next experiment, we examined how well our upper and lower bounds predict to an experimentally derived estimate of the actual PCR value. The actual PCR values were computed as follows: First create a a set of randomly generated distorted feature sets by using the distortion algorithm on the database models. We computed exactly 4 distorted copies of each model. We then run the recognition algorithm on the distorted feature sets to compute the resulting hypothesis for each. The PCR computed is the fraction of correctly recognized feature sets, meaning the ones whose resultant hypothesis matches the model from which the feature set was created.

In this preliminary study, we report results for the model database consisting of only 194 images of a T-72 tank in configuration #132. Results are shown in Figure 2. As can be seen, the results are not good in that

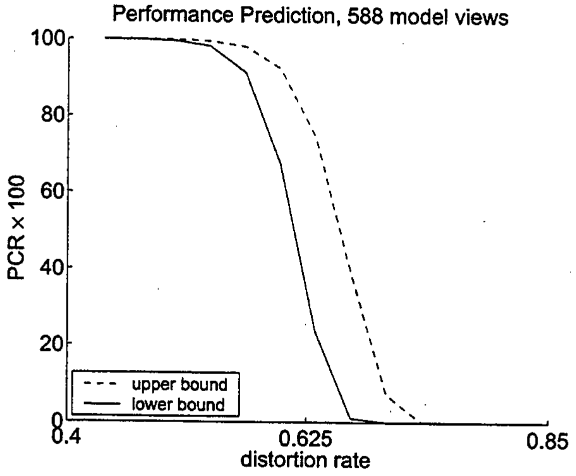


Figure 1. PCR results for prediction of upper and lower bounds.

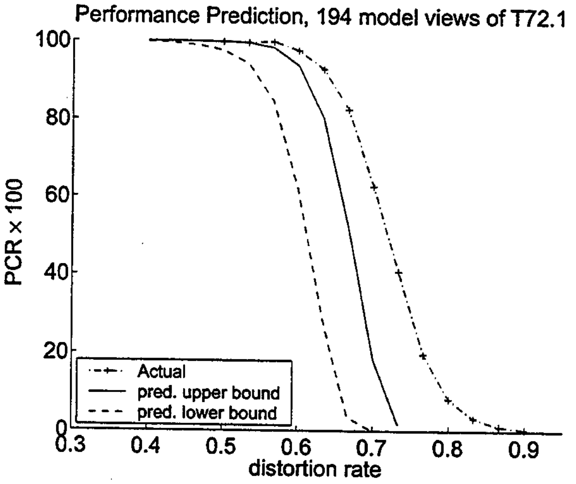


Figure 2. PCR prediction results compared to actual values.

the bounds are not that close to the actual value, and in fact the predicted upper bound is not even above the actual value.

After analyzing our method, we determined that the most significant source of error appears to be the approximation of $\Pr[V(\hat{M}, h_c) \leq V(\hat{M}, h)]$ with $\Pr[|M| - O \leq V(\hat{M}, h)]$, where h_c is the correct hypothesis. When the distortion rate is low, we expect $V(\hat{M}, h_c)$ to get about $|M| - O$ votes since without much added clutter, the correct hypothesis should not be getting many extra clutter votes incorrectly matched. However, as the distortion rates get large, the approximation gets less and less accurate as the larger amount of clutter makes it more likely to add in extra votes for the correct hypothesis. In fact, we can see that our bounds for PCR are getting worse as the distortion rate increases. It is also clear that the upper bound presented in Section 2.5 is not really an upper bound. The upper bound approximation (equation (4)) really needs to be greater than the previous upper bound equation (equation (3)), when in fact, it is less than that.

The two main problems with our above method are that the lower bound is not that close to the actual value, and the upper bound is completely inaccurate. One way of potentially fixing these problems is to simply try to approximate the actual value, rather than trying to get both upper and lower bounds. To this end, we also ran our final preliminary experiment, where we approximated $\Pr[V(\hat{M},h_c) > V(\hat{M},h)]$ with $\Pr[\mathrm{E}(V(\hat{M},h_c)) > V(\hat{M},h)]$, where h_c is the correct hypothesis, and E() denotes expected value. Here, we computed $\mathrm{E}(V(\hat{M},h_c))$ as $|M| - O + \lfloor \mathrm{E}(V(\hat{M},C)) \rfloor + 1$, where $V(\hat{M},C)$ is a random variable for the number of clutter votes that would be added when given a distorted model M and C clutter features. The overall approximate prediction value can be written as:

$$\prod_{h \in H} \Pr[\mathrm{E}((V\hat{M}),h_c) > V(\hat{M},h)].$$

This approximation result is shown in Figure 3, for both the 3D features (including the magnitude), and for 2D features where the magnitude is ignored.

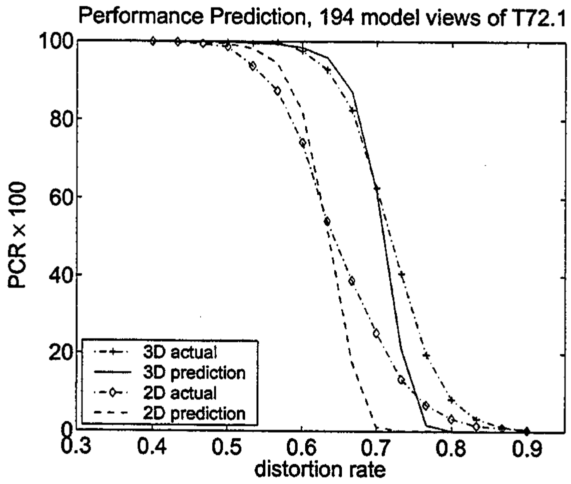


Figure 3. The new PCR prediction results compared to actual values, for both 2D (location only) and 3D (including magnitude) feature sets.

As can be seen, the predicted results are close to the actual values. Also note that when we add the third dimension of magnitude, We can have 10% more distortion than before while still maintaining the same PCR level as before.

4. CONCLUSIONS

We have presented a general model for predicting the bounds on performance of a wide variety of vote-based object recognition systems. We model features as a 2D location and a magnitude, and consider uncertainty (in both magnitude and location), occlusion, clutter, and model similarity as part of our performance prediction. We have done some preliminary evaluation of our general model to show its potential effectiveness.

Much, however, remains to be done in this work. More comprehensive tests on more sets of models need to be carried out. Furthermore, instead of only comparing to synthetically generated distorted data, performing tests on actual data distortion would help to validate our work. Furthermore, it would be interesting to use our model to predict the performance of vote-based schemes[3] and.[4] Comparing our predictions to these implemented recognition systems would be useful in showing that the sanitized, mathematically-described algorithms of vote-based recognition that we use actually are similar enough to the actually implemented algorithms and that our prediction can in fact predict the behavior of the real algorithms.

ACKNOWLEDGMENTS

This work was supported by grant F49620-01-1-0343. The contents of the information do not necessarily reflect the position or policy of the U.S. Government.

REFERENCES

1. M. Boshra and B. Bhanu, "Predicting performance of object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**, pp. 956–969, September 2000.
2. M. Boshra and B. Bhanu, "Predicting an upper bound on SAR ATR performance," *IEEE Transactions on Aerospace and Electronic Systems* **37**, pp. 876–888, July 2001.
3. G. Jones and B. Bhanu, "Recognition of articulated and occluded objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**, pp. 603–613, July 1999.
4. B. Bhanu and G. Jones, "Recognizing target variants and articulations in synthetic aperture radar images," *Optical Engineering* **39**, pp. 712–723, March 2000.